Bezdek, J.C., Chandrasekhar, R. and Attikiouzel, Y. (1998) A geometric approach to edge detection. IEEE Transactions on Fuzzy Systems, 6 (1). pp. 52-75.

A Geometric Approach to Edge Detection

James C. Bezdek, *Fellow, IEEE*, Ramachandran Chandrasekhar, *Member, IEEE*, and Yianni Attikiouzel, *Fellow, IEEE*

***Abstract*—This paper describes edge detection as a composition of four steps: conditioning, feature extraction, blending, and scaling. We examine the role of geometry in determining good features for edge detection and in setting parameters for functions to blend the features. We find that: 1) statistical features such as the range and standard deviation of window intensities can be as effective as more traditional features such as estimates of digital gradients; 2) blending functions that are roughly concave near the origin of feature space can provide visually better edge images than traditional choices such as the city-block and Euclidean norms; 3) geometric considerations can be used to specify the parameters of generalized logistic functions and Takagi–Sugeno input–output systems that yield a rich variety of edge images; and 4) understanding the geometry of the feature extraction and blending functions is the key to using models based on computational learning algorithms such as neural networks and fuzzy systems for edge detection. Edge images derived from a digitized mammogram are given to illustrate various facets of our approach.**

***Index Terms*— Edge detection, edge features, fuzzy systems, logistic functions, mammography, model-based training, Takagi–Sugeno model.**

I. Introduction

TRADITIONALLY, edge detection is regarded either as a convolution or correlation operation. For our purposes, it suffices to divide edge detectors into two broad groups based on the support of the convolution kernel. When the kernel has finite support, we get mask-based edge detectors such as the Sobel, Prewitt, Roberts, and Kirsch models [1]. A different class of models arises if the kernel has infinite support. For example, when the kernel is a two-dimensional (2-D) Gaussian distribution, we have the Marr–Hildreth [2] (Laplacian of the Gaussian) and Canny [3] models. More recently, Shen and Castan [4] proposed a 2-D symmetric exponential distribution for the kernel. There have also been many studies of edge detection with learning models that mimic one style or the other. This class includes, for example, both computational neural networks [5]–[10] and fuzzy reasoning systems [11]–[16].

The approach advocated in this article is limited to edge detection based on kernels with finite support. Our objective is to develop a common framework in which many approaches to edge detection can be understood and to illustrate its utility for designing both traditional and nontraditional (learning model) edge detectors. Section II establishes the notation and general architecture that underlies our approach. Sections III–VI discuss the four main operations we associate with edge detection (conditioning, feature extraction, blending, and scaling). Section V-D illustrates our model-based approach to training by discussing an optimal Takagi–Sugeno model for edge detection. Section VII contains examples with a digitized mammogram, which illustrate and compare various facets of our architecture. Finally, Section VIII contains our conclusions and a discussion about future research in edge detection.

Manuscript received May 11, 1996; revised September 22, 1997. J. C. Bezdek was supported by ONR Grant N00014-96-1-0642. R. Chandrasekhar was supported in part by an Australian OPRS scholarship and a university postgraduate award by UWA.

J. C. Bezdek is with the Department of Computer Science, University of West Florida, Pensacola, FL 32514 USA.

R. Chandrasekhar and Y. Attikiouzel are with the Center for Intelligent Information Processing Systems, Department of Electrical and Electronic Engineering, University of Western Australia, Nedlands, Perth, Western Australia, 6907 Australia.

Publisher Item Identifier S 1063-6706(98)01198-9.

II. The Elements of Edge Detection

We use the standard notation for *functions* from p-space to q-space $\mathbf{f} : \Re^p \mapsto \Re^q$. The *domain* and *range* of $\mathbf{f}$ are subsets of $\Re^p$ and $\Re^q$ denoted here as $\mathcal{D}_\mathbf{f}$ and $\mathcal{R}_\mathbf{f} = \mathbf{f}[\mathcal{D}_\mathbf{f}]$, respectively. The *graph* of $\mathbf{f}$ is $\mathcal{G}_\mathbf{f} = \{[\mathbf{x}, \mathbf{f}(\mathbf{x})]: \mathbf{x} \in \mathcal{D}_\mathbf{f}\} \subset \mathcal{D}_\mathbf{f} \times \mathcal{R}_\mathbf{f}$. For example, let $f: \Re \mapsto \Re^+$ be $f(x) = x^2$. If we restrict f to $[-1, 1]$, then $\mathcal{D}_\mathbf{f} = [-1, 1]$, $\mathcal{R}_\mathbf{f} = [0, 1]$, and $\mathcal{G}_\mathbf{f} = \{(x, x^2): x \in [-1, 1]\}$. A plot of $\mathcal{G}_\mathbf{f} \subset \Re \times \Re^+$ yields the familiar parabolic arc above the interval $[-1, 1]$. It is important to understand this construction, because learning models for edge detection can base the acquisition of training data on the geometry of the graph of a blending function.

Let $IJ = \{(i, j): i = 1, 2, \cdots, m;\ j = 1, 2, \cdots n\} \subset \Re^2$ denote a rectangular array of integers that specify (mn) *spatial locations* (pixel addresses). In what follows, we use ij as a short form for (i, j). Next, let $Q = \{q: q = 0, 1, \cdots, q_{\max}\} \subset \Re$ be the integers from zero to $q_{\max}$. Q is the set of *quantization (or gray) levels* for digitization of a *picture function* $\mathbf{p}$: $\Re^2 \mapsto \Re^N$. Integer N is the number of bands collocated in space and time that are measured by a sensor and $\mathbf{p}_{ij} = (p_{1,ij}, \cdots, p_{N,ij})$ is the vector of intensities located at pixel ij. For example, $N = 1$ for gray level images, $N = 3$ for most magnetic resonance (MR) images; $N = 6$ for coastal zone color scanner (CZCS) images, and so on. Confinement of $\mathbf{p}$ to the lattice IJ (done automatically by the digitizing scanner that realizes samples of $\mathbf{p}$) creates the $m \times n$ digital image $\mathbf{P}_{IJ}$. $\mathbf{P}_{IJ}$ is a *unispectral* image when $N = 1$ (we write P_{IJ} to indicate this); otherwise, it is *multispectral*. Our examples are confined to images P_{IJ} for which $q_{\max} = 255$.

$\mathbf{P}_{IJ}$ is a discrete subset of the graph of the picture function composed of the lattice IJ and the values of $\mathbf{p}$ on this lattice, $P_{IJ} \subset IJ \times Q^N \subset \mathcal{G}_\mathbf{p}$. More generally, it is advantageous to regard several images derived from $\mathbf{P}_{IJ}$ as subsets of the graph of some function defined on the lattice IJ. *A window*

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$$\leftrightarrow \mathbf{w} = (w_1,\ w_2,\ w_3,\ w_4,\ w_5,\ w_6,\ w_7,\ w_8,\ w_9)^T$$

Fig. 1. A 3 × 3 window in a unispectral image.

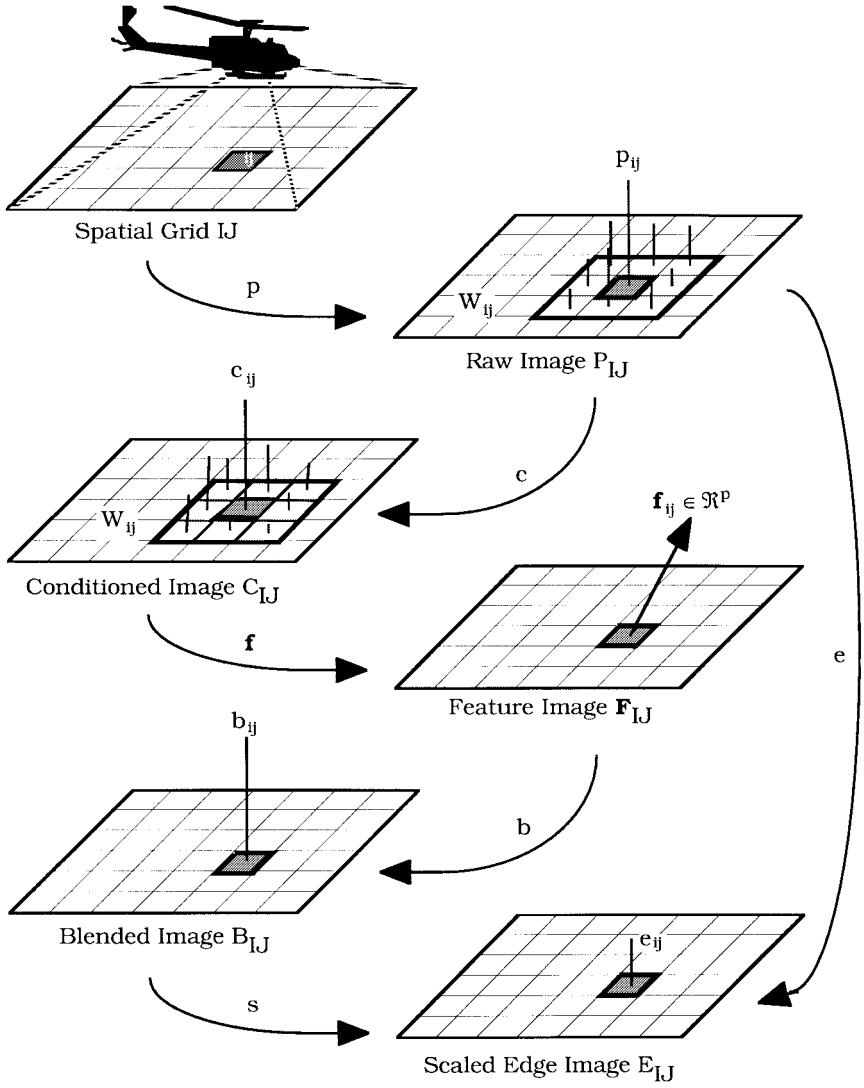


Fig. 2. The edge detection paradigm for $N = 1$.

$\mathbf{W}_{ij}$ related to pixel ij in $\mathbf{P}_{IJ}$ is a subset of $\mathbf{P}_{IJ}$, $\mathbf{W}_{ij} \subset P_{IJ} \subset \mathcal{G}_{\mathbf{p}}$. Thus, a window is a collection of addresses in the lattice IJ together with the intensities (or intensity for $N = 1$) at these addresses. The window as we have defined it is a subset of $\mathcal{G}_{\mathbf{p}}$ so the graph of the window is well defined. For our purposes it suffices to restrict $\mathbf{W}_{ij}$ to windows of *size* $m_{\mathbf{W}} \times n_{\mathbf{W}}$ *centered* at pixel ij, where $m_{\mathbf{W}}$ and $n_{\mathbf{W}}$ are odd integers. Our definition of windows includes the special case $\mathbf{W}_{ij} = ij$ (a pixel and its intensities are the window); we write W_{ij} when $N = 1$.

It is convenient to have a fixed indexing scheme for windows. Fig. 1 defines a correspondence between a 3 × 3 window W_{ij} and a sequentially labeled *window vector* $\mathbf{w} \in \Re^9$ of the intensities at the locations in the window. The center address in this window is ij, and it occupies position 5 in $\mathbf{w}$. For simplicity, we may use single subscripts for the pairs (i, j) in W_{ij}. We assume that the origin of spatial coordinates is at the upper left hand corner of W_{ij}, the horizontal x axis to the right, the vertical y axis downward.

Fig. 2 illustrates the architecture we use for edge detection. There will be special cases that need more (or less) components for a complete description and the extension (or compression) of our diagram for those cases will usually be obvious.

We view edge detection (and many other image processing operations such as segmentation, boundary analysis, etc.)

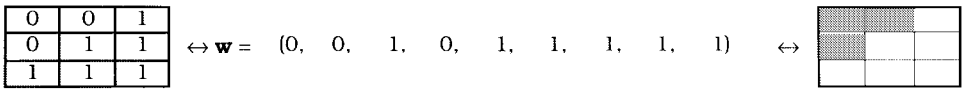


Fig. 3. Example of a binary-valued training vector.

as a sequence of operations that can be represented as a composition of functions. Careful specification of each step in the process greatly improves model understanding and, hence, optimization of model performance. Specifically, we denote the edge image as $E_{IJ} = e[P_{IJ}]$ so that $e = s \circ b \circ \mathbf{f} \circ c$. We regard the function e: $P_{IJ} \mapsto IJ \times Q$ as the *edge operator*. The four functions comprising e are

c: $IJ \times Q \mapsto IJ \times \Re$: $c[P_{IJ}] = C_{IJ}$
conditions (enhances) the raw sensor data P_{IJ}
$\mathbf{f}$: $IJ \times \Re \mapsto IJ \times \Re^p$: $\mathbf{f}[C_{IJ}] = \mathbf{F}_{IJ}$
extracts feature vectors in $\Re^p$ from C_{IJ}
b: $IJ \times \Re^p \mapsto IJ \times \Re$: $b[\mathbf{F}_{IJ}] = B_{IJ}$
blends components of feature vectors in $\mathbf{F}_{IJ}$
s: $IJ \times \Re \mapsto IJ \times Q$: $s[B_{IJ}] = E_{IJ}$
scales raw edge image B_{IJ} to get gray levels in Q.

Next, we introduce a small set of windows that provide a *basis* for modeling certain facets of edge detection. Fig. 3 depicts a specific 3×3 window vector $\mathbf{w}$ that has binary-valued components. Think of the value zero as corresponding to the gray level zero, and the value one as the gray level $q_{\max}$, which here is 255.

The center pixel in this window would most likely be designated as an edge pixel, since the window (visually) possesses an edge along its right diagonal. Restricting intensities to two levels in all nine locations yields $2^9 = 512$ possible binary-valued window vectors. We denote this set as B_{512} and call it an *edge detector basis set*

$$B_{512} = \{0, 1\}^9 = \underbrace{\{0, 1\} \times \{0, 1\} \times \cdots \times \{0, 1\}}_{9 \text{ times}}. \quad (1)$$

Our discussion of model-based training depends heavily on the notion of a basis such as B_{512}. However, this particular choice is but one of many that might be equally well suited to this task. We will show how to use B_{512} (or any set like it) to determine *input–output* (IO) training data for either a computational neural network or a fuzzy system such as the Takagi–Sugeno model.

III. Conditioning Functions (c)

The function c: $IJ \times Q \mapsto IJ \times \Re$: $c[P_{IJ}] = C_{IJ}$ includes many well-known procedures that are generally lumped together as image enhancement. Such procedures condition images that are unduly dark, bright, noisy, etc. to improve their utility as data to support answering some question related to the image. For example, one common pixel to pixel operation is *contrast enhancement*. Another, *histogram equalization*, is particularly attractive as a conditioning operation for edge detection. This procedure normalizes image intensities so that C_{IJ} has a linear cumulative histogram over Q. Since $\mathcal{D}_c \subset IJ \times Q$, the function c is well defined on individual pixels or on windows centered at them. Let n_k be the number of pixels in $m \times n$ image P_{IJ} with intensity q_k, $k = 0, 1, \cdots, q_{\max}$. The usual form of equalization is [17]

$$c_{ij} = c(p_{ij} = q_r) = q_{\max} \bullet \sum_{j=0}^{r} (n_j / mn). \quad (2)$$

We use a special form of histogram equalization in our examples. Let $\mathcal{H}$ and $\mathcal{C}$ be the histogram and cumulative histogram functions of P_{IJ}, respectively, and put

$$\mathcal{C}(0) = \mathcal{H}(0);\ \mathcal{C}(q) = \mathcal{C}(q-1) + \mathcal{H}(q)$$
$$q = 1, 2, \cdots 254; \mathcal{C}(255) = \text{mn}. \quad (3)$$

Now we define the equalized intensity at pixel (ij) as follows:

$$c_{ij} = 0 \text{ whenever } p_{ij} = 0 \quad (4a)$$

$$c_{ij} = \text{floor}\left[\frac{\mathcal{C}(q) \bullet (256 - 10^{-6})}{mn}\right] \text{ whenever } p_{ij} = q > 0. \quad (4b)$$

We use (4) to ensure that the range of c spans $Q = \{0, 1, \cdots, 255\}$ rather than a small subset of it. Generally, this might not be necessary, but in the application domain, we are particularly interested in (digital mammography), $\mathcal{C}(0) = \mathcal{H}(0)$ can be on the order of $0.2 \bullet mn$ (i.e., about 20% of the pixels are black). Moreover, this also provides contrast enhancement between pixels in P_{IJ} that have the values zero and one.

Conditioning operators defined on windows centered at ij include spatial filters (average, median, etc.) and unsharp masking, which is the subtraction of a low-pass filtered version of the image from the original. Conditioning is an important determinant in the quality of E_{IJ}. When c is used C_{IJ} becomes the digital image from which features are extracted. Henceforth, when we say picture function, we mean p if c is not used or $c \circ p$ if the original image is enhanced by any means before features are extracted from it.

IV. Feature Extraction Functions ($\mathbf{f}$)

The vector field $\mathbf{f}$: $IJ \times R \mapsto IJ \times \Re^p$: $\mathbf{f}[C_{IJ}] = \mathbf{F}_{IJ}$ is arguably the most important factor in the composite operator e. Since $\mathcal{D}_{\mathbf{f}} \subset IJ \times \Re$, $\mathbf{f}$ is well defined on individual pixels and on windows centered at them. In the context of edge detection, windows in C_{IJ} with size greater than 1×1 are almost always used for the p-vector of features extracted from W_{ij} that are attached to pixel ij in $\mathbf{F}_{IJ}$. The reason for this is that edges in the picture function are generally regarded as places where the surface $\mathcal{G}_p$ or $\mathcal{G}_c$ experiences large local distortion and data from more than one pixel is needed to detect this characteristic.

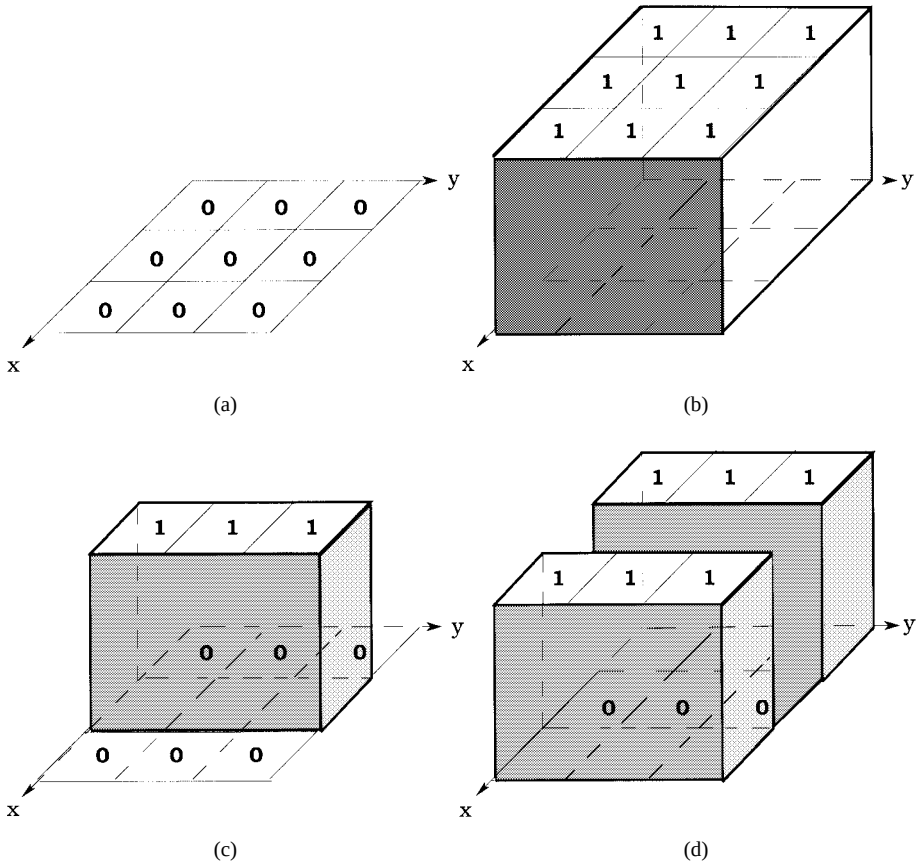


Fig. 4. Graphs of some windows from B_{512}.

Feature extraction functions valued in $\Re^2$ are usually designed to estimate some indicator of geometric behavior at edges in $\mathcal{G}_p$ or $\mathcal{G}_c$. The most obvious choice is an $\mathbf{f}$ that produces approximations to the gradient of the picture function because the gradient possesses two well-known geometric properties. At a point $\mathbf{x}_0 \in \Re^2$, $\nabla p(\mathbf{x}_0)$ points in the direction of maximum rate of increase of p and its magnitude (length in a chosen norm) gives the rate of change in this direction. At an edge, the magnitude of $\nabla p(\mathbf{x}_0)$ should be very large. However, the quality of numerical estimates of the gradient *at a pixel* in a digital image depends crucially on the resolution at which p is sampled. For example, if we assume (in orthogonal directions x and y) that $\Delta x = \Delta y = 1$, then the first-order forward, backward, and central differences for estimation of $\partial p/\partial x$ at cell 5 in Fig. 1 are, respectively, $(w_6 - w_5)$, $(w_5 - w_4)$, and $(w_6 - w_4)/2$. It is clear that abrupt changes in $\mathcal{G}_p$ or $\mathcal{G}_c$ (which correspond to visually perceptible edges) can be missed entirely using first-order differences unless image resolution is very high.

Derivatives are sensitive to noise and, because of this, many estimates of the gradient are combined with a smoothing operation. Well known *gradient-like* features that mitigate resolution and noise problems to some extent for 3×3 windows by using more information than first-order differences include the Sobel and Prewitt masks. Sometimes the masks are called operators; it is also common to regard them as filters, in which case the scalars associated with the mask are called filter coefficients.

TABLE I
SOBEL, PREWITT, AND RSD FEATURES FOR THE WINDOWS IN FIG. 4

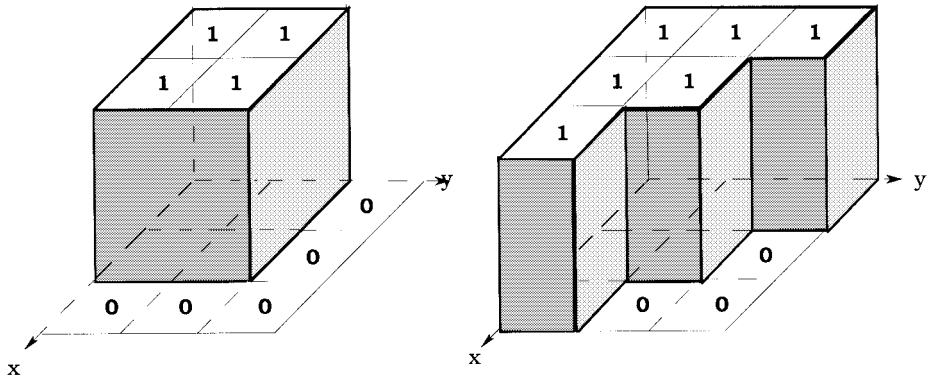
$\mathbf{w} \in \mathrm{B}_{512}$	$\mathrm{f}_{\mathrm{S}}(\mathbf{w})$	$\mathrm{f}_{\mathrm{P}}(\mathbf{w})$	$\mathrm{f}_{\mathrm{RSD}}(\mathbf{w})$
$\mathbf{w}_{\mathrm{A}}$	(0, 0)	(0, 0)	(0, 0)
$\mathbf{w}_{\mathrm{B}}$	(0, 0)	(0, 0)	(0, 0)
$\mathbf{w}_{\mathrm{C}}$	(0, 0)	(0, 0)	(1, 0.471)
$\mathbf{w}_{\mathrm{D}}$	(0, 0)	(0, 0)	(1, 0.471)
$\mathbf{w}_{\mathrm{E}}$	(-3, -3)	(-2, -2)	(1, 0.497)
$\mathbf{w}_{\mathrm{F}}$	(-3, -3)	(-2, -2)	(1, 0.471)

We feel it advantageous to represent the action of such masks or filters as feature extraction functions, because this frees us from the ideas that 1) they must be 2-D estimators of the digital gradient and, more importantly, 2) that they must produce features from digitally orthogonal subdomains of W_{ij} [as does any numerical estimate of $\nabla p(\mathbf{x}_0)$]. Using the representation of a 3×3 window vector $\mathbf{w}$ (as shown in Fig. 1) and letting h, v stand, respectively, for horizontal and vertical spatial directions, the Sobel feature extractor function is

$$\mathbf{f}_{\mathrm{S}}(\mathbf{w}) = [f_{\mathrm{Sh}}(\mathbf{w}),\ f_{\mathrm{Sv}}(\mathbf{w})]\text{: } \textbf{Sobel features} \tag{5a}$$

where

$$f_{\mathrm{Sh}}(\mathbf{w}) = (w_9 + 2w_8 + w_7) - (w_1 + 2w_2 + w_3) \tag{5b}$$



(e) (f)

Fig. 4. (*Continued*.) Graphs of some windows from B_{512}.

and

$$f_{\mathrm{Sv}}(\mathbf{w}) = (w_3 + 2w_6 + w_9) - (w_1 + 2w_4 + w_7). \quad (5c)$$

To emphasize that (5) is but one of many possible sets of features on which estimates of local surface behavior at this pixel can be based, we exhibit the Prewitt feature extractor for the same window

$$\mathbf{f}_{\mathrm{P}}(\mathbf{w}) = [f_{\mathrm{Ph}}(\mathbf{w}),\ f_{\mathrm{Pv}}(\mathbf{w})] : \textbf{Prewitt features} \quad (6a)$$

where

$$f_{\mathrm{Ph}}(\mathbf{w}) = (w_9 + w_8 + w_7) - (w_1 + w_2 + w_3) \quad (6b)$$

and

$$f_{\mathrm{Pv}}(\mathbf{w}) = (w_3 + w_6 + w_9) - (w_1 + w_4 + w_7). \quad (6c)$$

It is particularly easy to see the important geometric idea that underlies (5) or (6) by considering some extreme examples of B_{512} windows using the Sobel or Prewitt features. Fig. 4 shows six windows in B_{512}. The picture function cannot have vertical edges, as shown in Fig. 4. However, digitization makes this possible for the graphs of the digital picture functions whose intensities appear in the windows. The window vectors corresponding to views 4(a), 4(b), $\cdots$, 4(f) are called $\mathbf{w}_A$, $\mathbf{w}_B$, $\cdots$, $\mathbf{w}_F$. Seen in views 4(a) and 4(b) are the window vectors $\mathbf{w}_A = \mathbf{0}$ and $\mathbf{w}_B = \mathbf{1}$ in $\Re^9$. Columns 2 and 3 of Table I list the (transposed) feature vectors in $\Re^2$ obtained by applying the Sobel and Prewitt operators to the windows in Fig. 4.

These calculations reveal some interesting properties of the Sobel and Prewitt features. First, $\mathbf{w}_E$ and $\mathbf{w}_F$ have identical Sobel features, but $\mathbf{w}_F$ has a more visually apparent edge than $\mathbf{w}_E$. The same observation holds for the Prewitt features. Second and much more importantly, the Sobel and Prewitt features for the flat surfaces defined by $\mathbf{w}_A$ and $\mathbf{w}_B$ are the *same* as the features for the *digital butte and canyon* defined by $\mathbf{w}_C$ and $\mathbf{w}_D$. All four of these windows are represented by the zero vector in $\Re^2$! Thus, these two feature extractors cannot distinguish between members of B_{512} that have no edges and some that seem to have two. More generally, both of these feature extractors will produce the zero vector on *any* $\mathbf{w} \in B_{512}$ that has zero–one symmetry with respect to the center pixel. For example, both will produce $\mathbf{0}$ from $\mathbf{w} = (0, 1, 0, 1, 1, 1, 0, 1, 0)$, which has a cross centered on slot 5 as its digital graph. This is clear from the symmetric nature of the b and c parts of (5) and (6) with respect to the window.

The desire to discriminate between flat surfaces and the edge walls of buttes and canyons led us to investigate features with other geometric properties. For example, the *range* (r) and *standard deviation* (sd) of the intensities in $\mathbf{w}$ are useful for this purpose. The geometric meaning of r is clear: it is an order statistic that measures the maximum distortion among the intensities in $\mathbf{w}$. The standard deviation also has an obvious meaning in the present context: it measures how much variation occurs in the intensities in $\mathbf{w}$ (and, hence, in the patch of $\mathcal{G}_p$ or $\mathcal{G}_c$ of which $\mathbf{w}$ is a sample). For 3×3 windows

$$\mathbf{f}_{r\mathrm{sd}}(\mathbf{w}) = [f_r(\mathbf{w}),\ f_{\mathrm{sd}}(w)] : \textbf{ Range \& standard deviation (rsd) features} \quad (7a)$$

where

$$f_r(\mathbf{w}) = \underbrace{\max}_{1=1,\cdots,9} \{w_i\} - \underbrace{\min}_{1=1,\cdots,9} w_i \quad (7b)$$

and

$$f_{\mathrm{sd}}(\mathbf{w}) = \sqrt{\left(\sum_{i=1}^{9} w_i^2/9\right) - \left(\sum_{i=1}^{9} w_i/9\right)^2}. \quad (7c)$$

The last column of Table I lists the values of the rsd features for the six windows in Fig. 4. If and only if all pixels in $\mathbf{w}$ have the same value will $r = \mathrm{sd} = 0$. As you can see, $\mathbf{w}_A$ and $\mathbf{w}_B$ are again represented by the zero vector (as they should be), but $\mathbf{w}_C$ and $\mathbf{w}_D$ have equal nonzero feature vectors. This makes sense because the butte and canyon are cutaways of each other from the flat surface defined by $\mathbf{w}_B$. Comparing the features for $\mathbf{w}_E$ and $\mathbf{w}_F$ we see that, unlike the Sobel and Prewitt features, these two graphs have (very slightly) different rsd representations.

On the other hand, you can see in Table I that, just as for the Sobel and Prewitt masks, there will be cases where rsd features fail to distinguish between what are perhaps visually different edge situations. For example, rsd features are identical for windows $\mathbf{w}_C$, $\mathbf{w}_D$, and $\mathbf{w}_F$ in Fig. 4. In fact, there are only five different value pairs for $\mathbf{f}_{r\mathrm{sd}}(\mathbf{w})$ when $\mathbf{w} \in B_{512}$. For example, *every* $\mathbf{w} \in B_{512}$ with three or six ones has the rsd features (0, 0.471)! This seems like a disadvantage of rsd

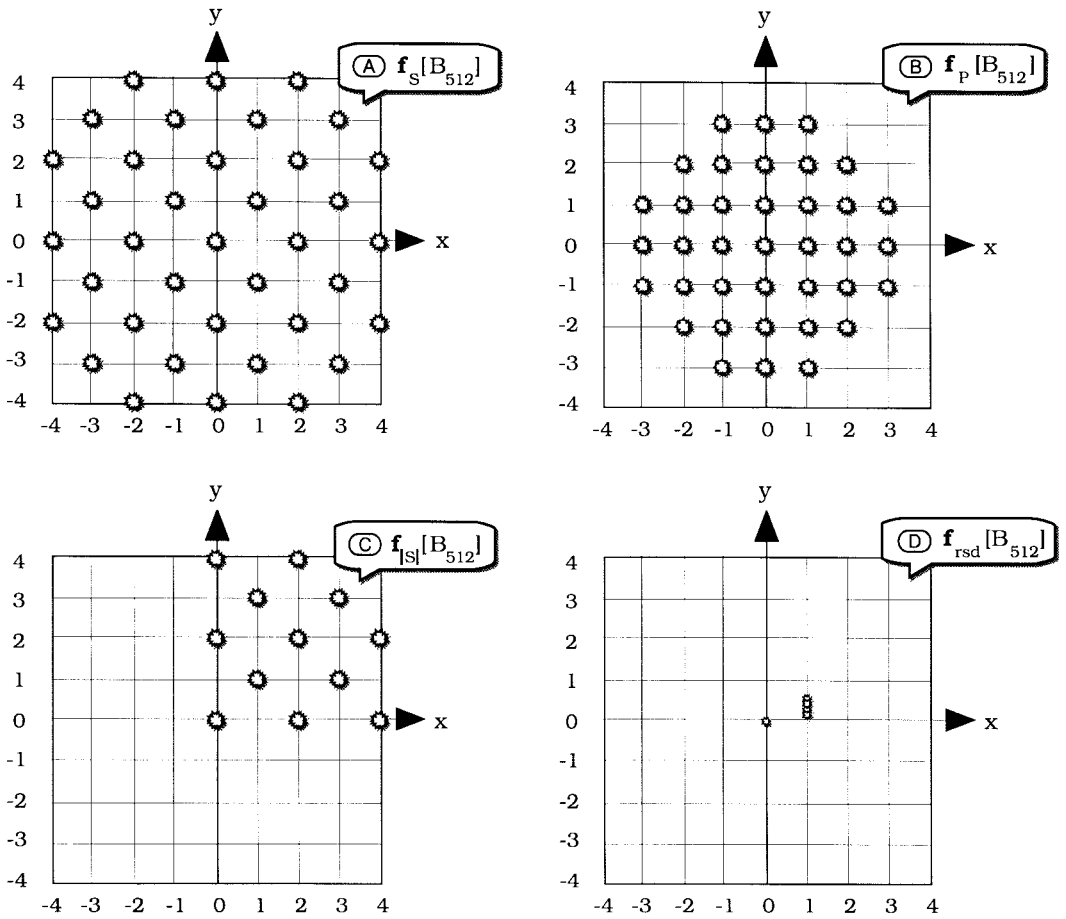


Fig. 5. $\mathbf{f}[B_{512}]$ for the Sobel, Prewitt, Absolute Sobel, and rSD feature extractors.

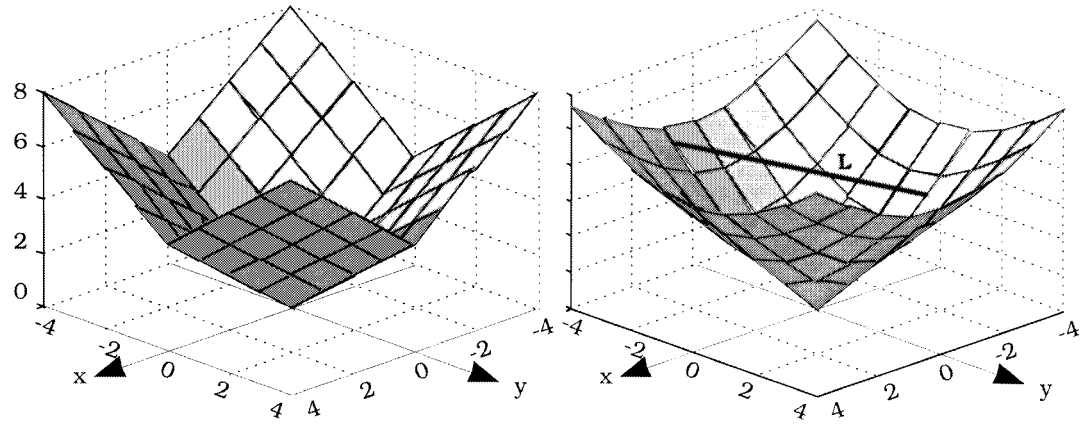
features, but remember, there are only two intensities in B_{512}; in a real image, the far larger range of intensities that appear in windows will impart more utility to these features than this example might suggest. It is not our intention here to advocate the rsd features. Rather, we encourage you to recognize that no matter what features you choose, there will be anomalies to overcome. The important points are to utilize your geometric intuition when choosing features and note that a model such as B_{512} makes this easier to do.

Generally, combinations and transformations of geometrically plausible features may be very effective for edge detection. For example, the quadruple $\mathbf{f}(\mathbf{w}) = [f_{\mathrm{Sh}}(\mathbf{w}), f_{\mathrm{Sv}}(\mathbf{w}), f_r(\mathbf{w}), f_{\mathrm{sd}}(\mathbf{w})] \in \Re^4$ is a set of four features that seem well suited for edge detection. The defects of $[f_{\mathrm{Sh}}(\mathbf{w}), f_{\mathrm{Sv}}(\mathbf{w})]$ may be mitigated by the strengths of $[f_r(\mathbf{w}), f_{\mathrm{sd}}(\mathbf{w})]$ and vice versa (and conversely, they might blur each other's good points too!). For example, the range and standard deviation are nonorthogonal on the digital grid but they are 1) rotationally invariant and 2) equations (7) are well-specified for all window sizes (even rectangular). On the other hand, gradient-like features are orthogonally oriented and have geometric properties that recommend them, but as the window size varies, it is unclear how to best estimate gradients. As a second example of feature transformations, we may use the absolute values or the squares of any of the feature sets in (5)–(7) as (new) features. This often affords algebraic economy and facilitates geometric visualization of the properties of the features.

We denote the image of B_{512} under feature extraction function $\mathbf{f}$ as $\mathbf{f}[B_{512}]$. This discrete set is in the range of $\mathbf{f}$ and so will be in the domain of $b\,\mathbf{f}$, $[B_{512}] \subset \mathcal{R}_{\mathbf{f}} = \mathcal{D}_b$. As an example, the values of $f_{\mathrm{Sh}}(\mathbf{w})$ and $f_{\mathrm{Sv}}(\mathbf{w})$ can individually range across the *integers* from -4 to 4 for $\mathbf{w}$ in B_{512}. However, some *combinations* of these integers cannot occur together because of the special (binary) nature of the 3×3 windows in B_{512}. Specifically, every point in $\mathbf{f}_{\mathrm{S}}[B_{512}]$ is a pair of integers (s, t) that satisfies the constraint $(s + t) \in \{0, \pm 2, \pm 4, \pm 6\}$.

It is important and helpful to understand what the image of B_{512} under various feature extractors looks like. To illustrate, let $\mathcal{A} = [-4, 4] \times [-4, 4]$. The set $\mathbf{f}_{\mathrm{S}}[B_{512}]$ consists of the 37 integer grid pairs in $\mathcal{A}$ shown in Fig. 5(a). $\mathbf{f}_{\mathrm{P}}[B_{512}]$ consists of the 37 integer grid pairs shown in view 5B; every point in $\mathbf{f}_{\mathrm{P}}[B_{512}]$ is a pair of integers (s, t) that satisfies the constraint $(s + t) \in \{0, \pm 1, \pm 2, \pm 3, \pm 4\}$. This is not surprising since $\mathbf{f}_{\mathrm{S}}$ and $\mathbf{f}_{\mathrm{P}}$ are linearly related.

Fig. 5(c) shows $\mathbf{f}_{|\mathrm{S}|}[B_{512}]$, where $\mathbf{f}_{|\mathrm{S}|}(\mathbf{w}) = (|f_{\mathrm{Sh}}(\mathbf{w})|, |f_{\mathrm{Sv}}(\mathbf{w})|)$ for $\mathbf{w} \in B_{512}$. Taking the absolute value of each component of the Sobel features simply folds the four quadrants of $\mathcal{A}$ onto the positive quadrant $\mathcal{A}^+ = [0, 4] \times [0, 4]$ shown in view 5(c), resulting in 12 integer pairs. This is the set of features we will use in the examples of Section VII.



(a) (b)

Fig. 6. The surfaces $\mathcal{G}_{||*||_1} = \{(\mathbf{x}, \|\mathbf{x}\|_1) \mid \mathbf{x} \in \mathcal{A}\}$ and $\mathcal{G}_{||*||_2} = \{(\mathbf{x}, \|\mathbf{x}\|_2) \mid \mathbf{x} \in \mathcal{A}\}$.

The set $\mathbf{f}_{rsd}[B_{512}]$ is shown in view 5(d). Letting n_1 and n_0 denote the number of ones and zeros in $\mathbf{w}$, it is easy to check that for $\mathbf{w} \in B_{512}$, $f_{sd}(\mathbf{w}) = \sqrt{n_1 n_0}/(n_1 n_0)$ and the range of $\mathbf{w}$ is either zero or one. Consequently, all windows with (zero or nine), (one or eight), (two or seven), (three or six), and (four or five) ones, map, respectively, to the feature pairs $(0, 0)^T$, $(1, 0.3143)^T$, $(1, 0.4157)^T$, $(1, 0.4714)^T$, and $(1, 0.4969)^T$. This paucity of input pairs may make the $r$sd features inadequate for *training* inputs to models that use learning algorithms as blending functions, but again, there are only two intensities in B_{512} compared to a far larger range of intensities in a real image.

Fig. 5 illustrates another important point about using a basis set such as B_{512} for training. As the function $\mathbf{f}$ changes, the coverage by $\mathbf{f}[B]$ of some base set such as A in Fig. 5 does too. This can have a dramatic effect on the approximation quality of the learning model being trained. It is difficult to see (visually) that $f_{rsd}[B_{512}]$ contains only five distinct pairs for all 512 windows in B_{512}: it is plotted on A at the same scale as the other three views so you can see how nonuniformly $\mathcal{A}$ is covered under different choices for $\mathbf{f}$.

V. Blending Functions (b)

Once features are chosen, we must pick the *blending function* b: $IJ \times \Re^p \mapsto IJ \times \Re$: $b[\mathbf{F}_{IJ}] = B_{IJ}$, which aggregates the information about edges possessed by $\mathbf{f}(\mathbf{w})$ for the purpose of edge detection. There are many types of blending functions. Of these, we discuss three parametric families: 1) *norms*, of which the two most common families are the inner product and Minkowski norms; 2) generalized logistic functions; and 3) computational learning models such as neural networks and fuzzy systems.

A. Norms

Let A be any positive definite $p \times p$ matrix. For $\mathbf{x} \in \Re^p$

$$\|\mathbf{x}\|_A = \sqrt{(\mathbf{x})^T A(\mathbf{x})} \tag{8}$$

is the *inner product norm* induced on $\Re^p$ by weight matrix A. Equation (8) defines an infinite family of norms on $\Re^p$ parametrized in A, the most important being the Euclidean norm which is induced by $A = I$, the $p \times p$ identity matrix

$$\|\mathbf{x}\|_I = \sqrt{(\mathbf{x})^T(\mathbf{x})}. \tag{9}$$

A second infinite family of norms that can be used for blending are the *Minkowski norms*, parametrized in t

$$\|\mathbf{x}\|_t = \left(\sum_{j=1}^{p} |x_j|^t\right)^{1/t}, \qquad t \geq 1. \tag{10}$$

The three most commonly used in practice are the one, two, and sup norms

$$\|\mathbf{x}\|_1 = \left(\sum_{j=1}^{p} |x_j|\right) \quad \text{City Block, } t = 1 \tag{11a}$$

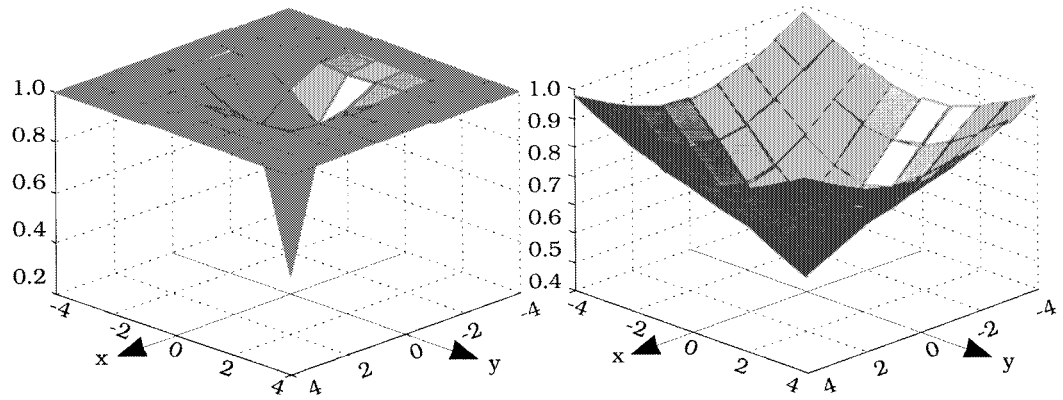
$$\|\mathbf{x}\|_2 = \left(\sum_{j=1}^{p} |x_j|^2\right)^{1/2} = \|\mathbf{x}\|_I \quad \text{Euclidean, } t = 2 \tag{11b}$$

and

$$\|\mathbf{x}\|_\infty = \max_{1 \leq j \leq p} \{|x_j|\} \quad \text{Sup or Max, } t \to \infty. \tag{11c}$$

For convenience, we let $\mathbf{x} = \mathbf{f}(\mathbf{w})$ and $b_{||*||}(\mathbf{x}) = b(\mathbf{x}; ||*||) = \|\mathbf{x}\|$ where $\|*\|$ is any norm on $\Re^p$. When needed, we may indicate the parameter of each of the families in (8) and (10) explicitly as $b(\mathbf{x}; \|*\|, A) = \|\mathbf{x}\|_A$ and $b(\mathbf{x}; \|*\|, t) = \|\mathbf{x}\|_t$, respectively. The Euclidean norm $b_{||*||_2}(\mathbf{x}) = \sqrt{\mathbf{x}^T\mathbf{x}}$ of the components of $\mathbf{f}(\mathbf{w})$ is often regarded as the standard blending function, but there is no reason *a priori* to prefer the Euclidean norm of, for example, the Sobel features $\mathbf{f}_S(\mathbf{w})$ as the best way to make Sobel edge images. The important point is that *any* norm can used to combine the components of $\mathbf{f}(\mathbf{w})$. Fig. 6(a) and (b) shows the graphs of the one and two norms on $\mathcal{A}$.

Viewing the blending function this way gives much insight into the properties of different edge operators. We see in Fig. 6(a) that the surface defined by the one norm is a cone with flat sides that are radially symmetric (in the sense of



(a) (b)

Fig. 7. The surfaces $\mathcal{G}_{b_L(\mathbf{x};\,||*||_2,\,\lambda,\,0.1)}$ on $\mathcal{A}$ for (a) $\lambda = 10$ and (b) $\lambda = 1$.

the one norm) about the z axis. Level sets of this surface are diamonds in the (x, y) plane centered at the origin. Similarly, the two norm generates a cone over $\mathcal{A}$ with circular level sets in the (x, y) plane. The graph of every norm on any feature space has a similar shape—it "holds water"—because all norms on $\Re^p$ are convex functions. Consequently, a line segment such as L in Fig. 6(b) connecting any pair of points in the graph of any norm will lie *in or above* the surface. Our experience is that this shape for the graph of the blending function is not particularly favorable for the detection of edges. We contend that blending functions that are *concave*, at least locally in the neighborhood of the origin, can lead to better edge images than norm functions. The next family of blending functions we discuss has this property.

B. Waterfall Functions

Blending functions that are *nearly concave* (instead of convex) *in the neighborhood of the origin* in feature space will be called *waterfall functions*, in analogy with the shape possessed by a profile view of a typical waterfall (i.e., they "shed water"). We do not give a mathematical definition of *nearly concave*, preferring instead to simply tell you what type of graphs we want. Generalized *logistic functions* in the p arguments of $\mathbf{x}$

$$b_L(\mathbf{x};\ ||*||,\ \lambda,\ \beta) = \frac{1}{1 + e^{-\lambda(||\mathbf{x}|| - \beta)}} \tag{12}$$

where $||*||$ is again *any* norm on $\Re^p$ and λ, β are real positive constants can be locally concave or convex near zero in $\Re^p$. b_L is an infinite family of three parameter blending functions whose general shape is affected more by (λ, β) than by the norm used in (12). For example, with the two norm and $\beta = 0.1$ fixed, varying λ in (12) produces a family of surfaces that range from an extremely "punctured" shape (large values of λ) such as is seen in Fig. 7(a) ($\lambda = 10$) that is locally concave near zero to surfaces that are locally convex near zero and, hence, resemble the graph of the inducing norm such as seen in Fig. 7(b) ($\lambda = 1$). Since $b_L(\mathbf{0}; ||*||, \lambda, \beta) = 1/(1 + e^{\lambda\beta}) > 0$ for any choice of parameters in (12), none of these functions takes the value zero at the origin. This is evident in Fig. 7, where you can see that the vertical scale in both plots starts above zero. In particular, $b_L(\mathbf{0}; ||*||_2, 10, 0.1) = 0.2689$ and $b_L(\mathbf{0}; ||*||_2, 1, 0.1) = 0.475$. Consequently, b_L cannot be zero on the window $\mathbf{w}_A$ in Fig. 4. Since the value $(b \circ \mathbf{f})(\mathbf{w})$ is interpreted as the extent to which $\mathbf{w}$ should be regarded as possessing an edge, this may seem counter-intuitive. However, we think that the response of b at the origin is less important than the change in response for $\mathbf{x}$'s near zero. Waterfall functions give large (more than linear) response to small changes near zero, whereas convex functions give lower increments in this neighborhood.

Fig. 8 shows the piece of the surface in Fig. 7(a) on $[0, 1] \times [0, 1]$. You can see why we call this a waterfall function. Notice two things. First, $b_L(\mathbf{x};\ ||*||_2,\ 10,\ 0.1)$ has essentially attained its asymptotic limit of one for vectors $\mathbf{x} = (x, y)^T$ lying beyond the circle $x^2 + y^2 = (0.5)^2 = 0.25$. For example, the value at $(0.5,\ 0.0)^T$ is 0.982, so only feature vectors near the origin will provide sharp differential edge response. Second, this surface appears to be concave along the axes of its domain and convex perpendicular to the line $y = x$ near the origin, so it is neither convex nor concave in this region. This is not by intention, but we point it out to make sure that you see the important point about waterfall functions: we think that $\Delta b = b(\mathbf{x} + \Delta\mathbf{x}) - b(\mathbf{x})$ should be large when $\mathbf{x}$ is close to zero and experiences a relatively small change $\Delta\mathbf{x}$.

C. Learning Models

The last type of blending functions discussed here are *computational transformations* (input–output systems) such as *neural-like networks* [18] and the *Takagi–Sugeno* (TS) fuzzy reasoning model [19]. Both of these schemes are well known for their ability to provide good approximations to rather arbitrary nonlinear vector fields. In the present context, they can be used effectively to blend vectorial inputs such as $\mathbf{f}(\mathbf{w})$ in $\Re^p$, and can be either trained or subjectively configured to produce good edge values at their output layers. We denote these two cases as b_{NN} and b_{TS}. Learning models require training data, and in particular, we need to know what "good edge output values" are for inputs such as $\mathbf{f}(\mathbf{w})$. To our knowledge only two previous studies use model-based training data (viz., B_{512}) for an edge detecting feed forward neural network b_{NN} [10] and for the TS model b_{TS} [16]. Here, we extend this idea to a much more general scheme that depends

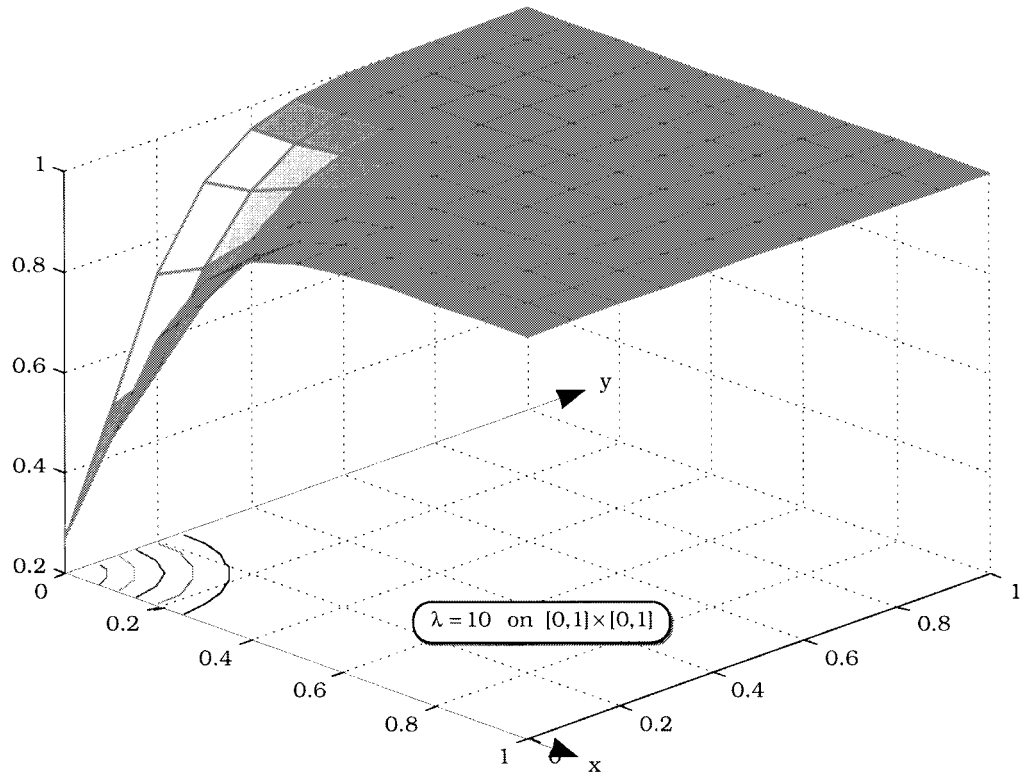


Fig. 8. The surface generated by b_L $(\mathbf{x};\ \|*\|_2, 10, 0.1)$ on [0, 1] × [0, 1].

explicitly on knowledge of the graph of the *desired* blending function. Specifically, $\{b[\mathbf{f}(\mathbf{w})]\colon\ \mathbf{w} \in B_{512}\}$ are the target output scores on the training inputs $\mathbf{w} \in B_{512}$. The desired (target) outputs are found by specifying $\mathbf{f}$ and b and then applying $b \circ \mathbf{f}$ to B_{512}. This gives a set of IO pairs, say $T_{\mathrm{IO}} = \{[\mathbf{w}, b \circ \mathbf{f}(\mathbf{w})]\colon\ \mathbf{w} \in B_{512}\}$ that can be used to train computational learning models for edge detection.

More generally, we let $T_{\mathrm{IO}}(B, \mathbf{f}, b) = \{[\mathbf{w}, b \circ \mathbf{f}(\mathbf{w})]\colon\ \mathbf{w} \in B\}$ to emphasize that there are three components to building T_{IO}: 1) the *model basis set* B; 2) the *feature extraction function* $\mathbf{f}$; and 3) the *blending function* b. Changing *any* of these three will change $T_{\mathrm{IO}}(B, \mathbf{f}, b)$ and also the edge detector subsequently trained with it. We expect the trained mapping to detect edges because 1) the window vectors $\mathbf{w} \in B$ are chosen selectively to reveal edges; 2) the features $\mathbf{f}(\mathbf{w})$ are chosen because they are related to geometric properties of edges; and 3) the blending function (b) is chosen so that its graph has a shape that is maximally responsive to the perception of edges.

Fig. 9 illustrates the construction of T_{IO}. The model basis is a box B containing the chosen set of window vectors $\mathbf{w} \in B$. The horizontal plane represents $\Re^p$, the chosen feature space. Choosing $\mathbf{f}$ determines the dimension p, and knowing the set of gray levels Q fixes $\mathcal{R}_{\mathbf{f}} \subset \mathcal{D}_b$. The training inputs $\mathbf{f}[B]$ will be a discrete subset in the domain of b, $\mathbf{f}[B] \subset \mathcal{R}_{\mathbf{f}} \subset \mathcal{D}_b$, as shown in Fig. 9. For example, any of the discrete sets $\mathbf{f}[B_{512}]$ (or combinations thereof) shown in Fig. 5 could appear as $\mathbf{f}[B]$ in Fig. 9. The target outputs are simply the values of b on $\mathbf{f}[B]$. It is here that knowledge of the desired shape of the blending function comes into play. Analytically expressed blending functions such as norms or generalized logistic functions are simply evaluated on $\mathbf{f}[B]$. Alternatively, *any* method of specification of the values $\{b \circ \mathbf{f}(\mathbf{w})\}$ completes $T_{\mathrm{IO}}(B, \mathbf{f}, b)$. (Some authors have scored windows completely "by eye" and have reported reasonable results [7].) After training, b_{NN} or b_{TS} becomes the blending function.

If we wanted to approximate the surface $\mathcal{G}_b$ over $\mathcal{D}_b$, we would probably choose a nicely arranged lattice of base points in $\mathcal{R}_{\mathbf{f}} \subset \Re^p$. Typically, such a lattice is rectangular and grid spacing is chosen small enough to insure that the training data provide a reasonable sample for fitting a surface to the points chosen. For edge detection, however, the input training points are chosen as points in $\mathcal{R}_{\mathbf{f}}$ that possess edge information; that is, points in $\mathbf{f}[B]$. Fig. 5 shows that these points may or may not cover the domain of b in a nice way from the standpoint of numerical analysis or from the standpoint of training and testing in pattern recognition. But from the standpoint of edge detection, we believe that very few points are actually needed, provided they are related to edge geometry, the shape of the blending surface is correct and at least some of the points are concentrated near the origin.

D. The Takagi–Sugeno (TS) Model

As a specific example of the ideas in Section V-C, we discuss the TS model, which is a fuzzy rule-based inference engine that can be used for function approximation. The multiple-input single-output (MISO) case is summarized in Fig. 10. In step ①, $\mathbf{x} = (x_1, x_2, \cdots, x_p)^T$ in $\Re^p$ is an input vector.

Step ② begins with the identification of the numerical range of each input variable. For each k, the numerical domain $\mathcal{D}_k$ is found and associated with a *linguistic variable* $\mathcal{L}_k$ that provides a semantic description of (r) subdomains of $\mathcal{D}_k$.

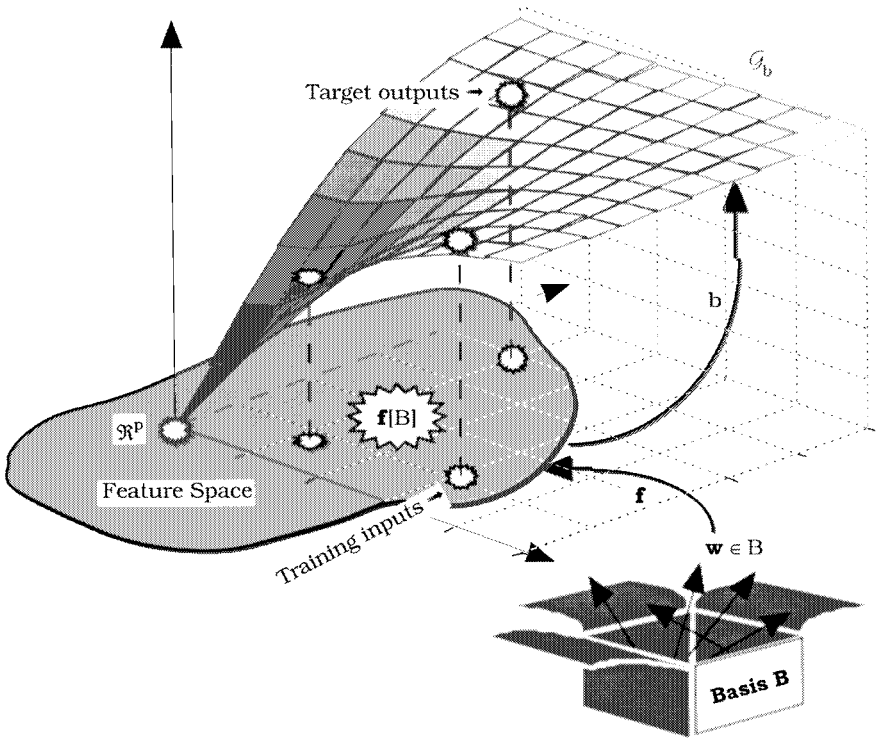


Fig. 9. The three elements of $T_{IO}(B, \mathbf{f}, b)$.

❶ Input	❷ Fuzzify	❸ RB: LHS ⇒ RHS	❹ Defuzzify ⇒ Output
$\mathbf{x} \in \Re^p$ $\begin{pmatrix} x_1 \\ \vdots \\ x_k \\ \vdots \\ x_p \end{pmatrix}$	$m_{11} \cdots m_{1r}$ $D_1 \leftrightarrow \mathcal{L}_1$ $m_{k1} \cdots m_{kr}$ $D_k \leftrightarrow \mathcal{L}_k$ $m_{p1} \cdots m_{pr}$ $D_p \leftrightarrow \mathcal{L}_p$	$\alpha_1(\mathbf{x}) = \mathcal{T}(\mathbf{m}^1(\mathbf{x})) \Rightarrow \mathbf{u}_1(\mathbf{x})$ $\alpha_q(\mathbf{x}) = \mathcal{T}(\mathbf{m}^q(\mathbf{x})) \Rightarrow \mathbf{u}_q(\mathbf{x})$ $\alpha_M(\mathbf{x}) = \mathcal{T}(\mathbf{m}^M(\mathbf{x})) \Rightarrow \mathbf{u}_M(\mathbf{x})$	$\mathbf{u} \in \Re^s$ $\mathbf{u}(\mathbf{x}) = \underset{q=1}{\overset{M}{\Theta}} \left(\alpha_q(\mathbf{x}), \mathbf{u}_q(\mathbf{x})\right)$

Fig. 10. The general architecture of the TS model.

The number (r) is called the *granularity* of $\mathcal{L}_k$; generally, r can be a function of k. The jth subdomain of $\mathcal{L}_k$ represents a *linguistic value*, say l_{kj}. This linguistic value is in turn represented by a fuzzy membership function m_{kj}: $\mathcal{D}_k \mapsto [0, 1]$. In Fig. 10, the membership functions all have symmetric triangular graphs, but again, this need not be the case. The set $\{m_{kj}: 1 \le j \le r\}$ is called the *linguistic termset* associated with variable k, $1 \le k \le p$. Step ② is often referred to as *fuzzification* of the input domain.

Step ③ comprises the reasoning mechanism. The left-hand side (LHS) and right-hand side (RHS) of the *rule base* (RB) are composed of M rules $\{R_q\}$ that operate on $\mathbf{x}$ and take the general form

$$R_1: \alpha_1(\mathbf{x}) = \mathcal{T}[\mathbf{m}^1(\mathbf{x})] = \mathcal{T}\left[\underbrace{m_{1j_1}(x_1), \cdots, m_{kj_k}(x_k), \cdots, m_{pj_p}(x_p)}_{\mathbf{m}^1(\mathbf{x}) \in \Re^p}\right] \Rightarrow u_1(\mathbf{x}); \quad (13a)$$

$$R_q: \alpha_q(\mathbf{x}) = \mathcal{T}[\mathbf{m}^q(\mathbf{x})] = \mathcal{T}\left[\underbrace{m_{1k_1}(x_1), \cdots, m_{kt_k}(x_k), \cdots, m_{pk_p}(x_p)}_{\mathbf{m}^q(\mathbf{x}) \in \Re^p}\right] \Rightarrow u_q(\mathbf{x}); \quad (13b)$$

$$R_M: \alpha_M(\mathbf{x}) = \mathcal{T}(\mathbf{m}^M(\mathbf{x})] = \mathcal{T}\left[\underbrace{m_{1i_1}(x_1), \cdots, m_{ki_k}(x_k), \cdots, m_{pi_p}(x_p)}_{\mathbf{m}^M(\mathbf{x}) \in \Re^p}\right] \Rightarrow u_M(\mathbf{x}). \quad (13c)$$

In (13b), $\alpha_q(\mathbf{x}) = \mathcal{T}[\mathbf{m}^q(\mathbf{x})]$ is called the *firing strength* of rule q: $1 \le q \le M$. Here, $\mathcal{T}$ is any $\mathcal{T}$-norm (intersection) operator [20] on [0, 1] × [0, 1]. Our notation is a little sloppy, because $\mathbf{m}^q(\mathbf{x})$ is not the value of a fixed vector field $\mathbf{m}^q$ on $\mathbf{x}$. Instead, the membership functions that yield the p arguments of $\mathcal{T}$ depend on the input $\mathbf{x}$. In other words,

different membership functions among the $\{m_{kj}\}$ will be used as $\mathbf{x}$ runs through its domain.

The output functions $\{u_q\colon \Re^p \mapsto \Re\}$ comprise the RHS of the rule base. Each u_q has a functional form (e.g., linear, affine, quadratic, polynomial, trigonometric, transcendental, power, etc.) specified by the user. It is common, but not necessary, to require that all the u_q's have the same functional form. Step ④ produces the numerical output $u(\mathbf{x})$ by combining the firing strengths and outputs of the individual rules with some aggregation operator Θ. Usually, Θ is taken as the weighted sum

$$u(\mathbf{x}) = \sum_{k=1}^{M} \alpha_k(\mathbf{x}) u_k(\mathbf{x}) / \sum_{j=1}^{M} \alpha_j(\mathbf{x}). \tag{14}$$

If any component in rule q (say the kth) is zero; then $m_{ks_k}(x_k) = 0 \Rightarrow \alpha_q(\mathbf{x}) = \mathcal{T}(\mathbf{m}^q(\mathbf{x})] = 0$. This means that a given input vector $\mathbf{x}$ in $\Re^p$ will probably never fire all of the rules in (13)—most of the $\alpha_q(\mathbf{x})s$ will be zero. If care is taken during fuzzification, it will never happen that *all* of the firing strengths are zero. Consequently, the system is well defined and can be constrained to have an output that always lies in a specified range $[u_{\min}, u_{\max}]$.

We can summarize (13) and (14) succinctly by noting that $TS\colon \Re^p \mapsto \Re$ is simply a mapping. This is entirely analogous to a computational neural network in that the function TS is specified by a computer program (as opposed to a function in closed form). Analogous to Kolmogorov's theorem for three-layer feed forward neural networks, there are many "universal approximation" theorems that give conditions under which mapping TS exactly represents continuous functions on compact subsets of $\Re^p$ [20].

Suppose that each u_q in (13) is parametrized by an unknown set of parameters $\mathbf{a}_q$ in a parameter space $\mathcal{W}_q$. Assuming that all other choices and parameters for the TS model in Fig. 10 are fixed, completion of the model means acquiring "good values" for the parameters $\{\mathbf{a}_q\colon 1 \le q \le M\}$. There are many ways to do this, partially dependent on the form of the output functions. When the u_q's are all affine (linear plus a constant), we call the system a *first-order* TS model; each output function can be written as a Euclidean dot product in $\Re^{p+1}$, $u_q(\mathbf{x}) = \langle \mathbf{a}'_q, \mathbf{x}' \rangle_{\Re^{p+1}}$, where $\mathbf{x}' = (1, x_1, \cdots, x_p)^T$ and $\mathbf{a}'_q = (a_{0q}, a_{1q}, \cdots, a_{pq})^T$. In this case, Takagi and Sugeno [19] show how to find a minimizing least-squared error (LSE) solution to a (possibly rectangular) linear system formed by submitting n_{T} IO pairs $\{[\mathbf{x}_k, u(\mathbf{x}_k)]\}$; $1 \le k \le n_{\mathrm{T}}\}$, to the TS model. This method uses the psuedo-inverse to find the LSE solution and, as n_{T} grows, it becomes more and more intractable numerically because the matrix of coefficients is usually quite sparse (not many rules fire for each input $\mathbf{x}_k$). Alternatives to this method that generalize well to more complicated types of output functions include gradient descent [21] and neural networks [22].

For some feature extraction and blending functions, it is possible to solve the LSE problem associated with the TS model analytically. Specifically, in [16] a 12-rule TS blending function, say b_{TS12}, *interpolates* the training data $\mathbf{T}_{\mathrm{IO}}(B_{512}, \mathbf{f}_{|S|}, b_{\|*\|_1})$, which has also been used to train a feed forward neural network edge detector having nine input nodes, one output node, and two hidden layers of seven and two nodes [10]. The resultant blending function b_{NN} was reasonably successful at detecting edges in various real images. Fig. 11 shows the linguistic termset and its membership functions for $x = |f_{\mathrm{Sh}}(\mathbf{w})|$. The y coordinate $y = |f_{\mathrm{Sv}}(\mathbf{w})|$ used the same set of functions. There are $r = 5$ membership functions over each of $\mathcal{D}_1$ and $\mathcal{D}_2$. The linguistic variable for both inputs is "quantity," and the linguistic values taken by the inputs are: *low* = L; *medium-low* = ML; *medium* = M; *medium-high* = MH; and *high* = H.

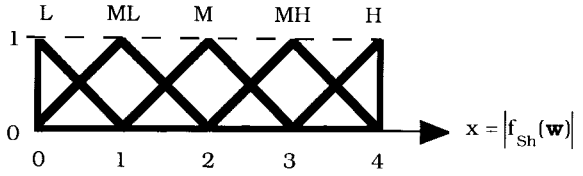


Fig. 11. Membership functions on $x = |f_{\mathrm{Sh}}(\mathbf{w})|$ for the 12-rule blending function b_{TS12}.

Each of the membership functions for x and y shown in Fig. 11 has the convenient representation

$$m_{kj}(z) = \max\{0, 1 - s_{kj}|c_{kj} - z|\}, \qquad k = 1, 2, \quad j = 1, 2, 3, 4, 5 \tag{15}$$

where s_{kj} is the positive slope of the leading edge line, and c_{kj} is the center of the nonzero portion of the graph (the point at which $m_{kj}(z) = 1$). For the case shown, the slopes are all one and the center of each function is set at one of the integers in D_k. For example, for a given input vector $\mathbf{x} = (x, y)^T$, the linguistic statement "x is medium" is represented by the value $m_M(x) = m_{13}(x) = \max\{0, 1 - |2 - x|\}$.

E. Using the Takagi–Sugeno Model without Training

Now we discuss *subjective design* as done in [11]–[15]. First, choose two semantic terms and membership functions along the x and y axes. The two membership functions for each input variable must still cover the numerical domain of the chosen features, which is [0, 4] for $(\mathbf{x}) = (x, y)^T$ where $x = |f_{\mathrm{Sh}}(\mathbf{w})|$ and $y = |f_{\mathrm{Sv}}(\mathbf{w})|$.

Fig. 12 shows membership functions for "low" and "high" (on x and y) for a four-rule TS model that represents the blending function we call b_{TS4}. Rather than *train* this model with some $\mathbf{T}_{\mathrm{IO}}(B, \mathbf{f}, b)$, we *specify* output functions for b_{TS4} that make it a four parameter family of models. Let $\tau, \chi, \gamma, \omega \in \Re^+$ and define the rules as

$$\text{R1. If } x = L \text{ and } y = L \Rightarrow u_1(\mathbf{x}) = x^\tau + y^\tau \tag{16a}$$

$$\text{R2. If } x = L \text{ and } y = H \Rightarrow u_2(\mathbf{x}) = \chi \tag{16b}$$

$$\text{R3. If } x = H \text{ and } y = L \Rightarrow u_3(\mathbf{x}) = \gamma \tag{16c}$$

$$\text{R4. If } x = H \text{ and } y = H \Rightarrow u_4(\mathbf{x}) = \omega. \tag{16d}$$

System (16), with the functions shown in Fig. 12, is a regular first-order TS model when $\tau = 1$; and otherwise, it is regular but neither linear nor polynomial (unless τ is an integer). Now we choose the $\mathcal{T}$ norm as the product of its arguments, $\mathcal{T}(a, b) = ab$. The functions in Fig. 12 satisfy

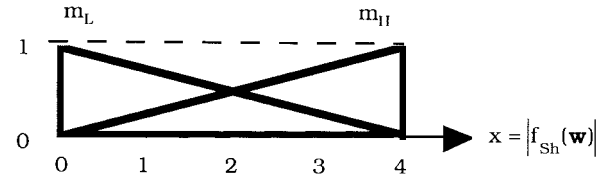


Fig. 12. Membership functions for b_{TS4}.

the relation $m_L(x) + m_H(x) = 1$ so $m_H(x) = 1 - m_L(x)$ for x in [0, 4], and similarly for the y variable. Since m_L and m_H are the same for x and y, the rules in (16) can all be written in terms of a single membership function $m(z) = \max\{0, 1 - |z|/4\}$ where z can be x or y in [0, 4]. The firing strengths shown in (13) for the four rules in (16) become

$$\text{R1. } \alpha_1(\mathbf{x}) = m(x) \bullet m(y) \tag{17a}$$

$$\text{R2. } \alpha_2(\mathbf{x}) = m(x) \bullet [1 - m(y)] \tag{17b}$$

$$\text{R3. } \alpha_3(\mathbf{x}) = [1 - m(x)] \bullet m(y) \tag{17c}$$

$$\text{R4. } \alpha_4(\mathbf{x}) = [1 - m(x)] \bullet [1 - m(y)]. \tag{17d}$$

Adding the four functions in (17) gives the denominator of (14), which in this special case is always one, $\sum_{j=1}^{4} \alpha_j(\mathbf{x}) = 1$. Using this, substitution of (17) into (14) yields an explicit formula for the output of the parametric family of four-rule TS models in (16)

$$\begin{aligned} b_{\mathrm{TS4}}(\mathbf{x};\, \tau, \chi, \gamma, \omega) = {} & m(x)m(y)[x^\tau + y^\tau + \omega - \chi - \gamma] \\ & + m(x)[\chi - \omega] + m(y)[\gamma - \omega] + \omega \end{aligned} \tag{18a}$$

where

$$\begin{aligned} m(z) &= \max\{0, 1 - |z|/4\} \\ z &= x \text{ or } y \in [0, 4]. \end{aligned} \tag{18b}$$

This is a particularly simple fuzzy system as its output can be computed directly with (18) rather than by approximate reasoning as shown Fig. 10. (See [30]–[32] for conditions under which TS systems can be reduced to explicit formulas.) Since $m(0) = 1$ and $m(4) = 0$, we can compute boundary conditions from (18a) at the four corners of the domain [0, 4] $\times$ [0, 4]

$$b_{\mathrm{TS4}}[(0, 0);\, \tau, \lambda, \beta, \omega] = 0 \tag{19a}$$

$$b_{\mathrm{TS4}}[(0, 4);\, \tau, \chi, \gamma, \omega] = \chi \tag{19b}$$

$$b_{\mathrm{TS4}}[(4, 0);\, \tau, \chi, \gamma, \omega] = \gamma \tag{19c}$$

$$b_{\mathrm{TS4}}[(4, 4);\, \tau, \lambda, \beta, \omega] = \omega. \tag{19d}$$

This shows that the constants χ, γ, and ω on the RHS of rules 2, 3, and 4 in (16) simply fix the height of b_{TS4} at the corresponding (nonzero) corners of its domain. Judicious choices of χ, γ, and ω might eliminate the necessity to scale Q to, e.g., [0, 4], as done here. It would be unusual not to specify $\chi = \gamma$ as this would destroy symmetry of the surface with respect to the plane $\{x = y, z = z\}$ in $\Re^3$ for features such as $f_{\mathrm{Sh}}(\mathbf{w})$ and $f_{\mathrm{Sv}}(\mathbf{w})$. On the other hand, when using features such as $f_r(\mathbf{w})$ and $f_{\mathrm{sd}}(\mathbf{w})$, which are not symmetrically matched, this flexibility might be turned to good advantage.

Rule 1 is the critical rule in determining the shape of the graph of b_{TS4}. The output function $u_1(\mathbf{x}) = x^\tau + y^\tau$ controls the shape of the blending surface in the neighborhood of $\mathbf{0}$. In fact, $x^\tau + y^\tau$ is just the τth power of the τ norm of $\mathbf{x}$ (when $\tau \geq 1$) shown in (10) and its values will dominate (18a) near $\mathbf{0}$ since $m(x)$ and $m(y)$ will both be close to one there.

In Fig. 13, $\chi = \gamma = 2$ and $\omega = 3$. For all x, y in (0, 1), choosing $\tau > 1$ makes $x^\tau + y^\tau$ smaller than $x + y$ and b_{TS4} is locally convex near $\mathbf{0}$, as illustrated in Fig. 13(a) where $\tau = 4$. For $\tau < 1$, the reverse effect produces local concavity for b_{TS4} near $\mathbf{0}$, as shown in Fig. 13(c), where $\tau = 1/4$. The transition case shown in Fig. 13(b) is for $\tau = 1$. This surface is concave along the axes and is essentially flat (neither concave or convex) close to $\mathbf{0}$. Fig. 13(a) shows that very simple TS models are capable of modeling highly nonlinear phenomena.

VI. Scaling Functions (s)

The last function shown in Fig. 2 is s: $IJ \times \Re \mapsto IJ \times Q$: $s[B_{IJ}] = E_{IJ}$, which simply scales the intensities in B_{IJ} so that the resultant edge image E_{IJ} has intensities in the original (or some other desired) set Q of gray levels that are associated with P_{IJ}. Here we describe a method of dynamic scaling used for the experiments in the next section.

The input image used in Section VII (both before and after histogram equalization) is a "byte image" with integer pixel values in $Q = \{0, 1, \cdots, 255\}$. All working images (holding the feature vectors blended in some way) are "float images," so each pixel value $b_{ij} \in B_{IJ}$ is a signed real number. For the purposes of display and storage, float images were reconverted to byte images using the following form of *dynamic scaling*

$$b_{\max} = \underbrace{\max}_{i,j}\{b_{ij}\}. \tag{20a}$$

$$b_{\min} = \underbrace{\min}_{i,j}\{b_{ij}\}. \tag{20b}$$

$$e_{ij} = \text{floor } [255 \bullet (b_{ij} - b_{\min})/(b_{\max} - b_{\min})]. \tag{20c}$$

$$\text{If } e_{ij} < 0 \text{ set } e_{ij} = 0; \quad \text{If } e_{ij} > 255 \text{ set } e_{ij} = 255. \tag{20d}$$

The clipping operation in (20d) is self-explanatory. Taken together, (20a)–(d) comprise the function s that we use for scaling. This operation is different than and independent of normalization factors explained at the beginning of Section VII that may be applied to the features $\mathbf{f}(\mathbf{w})$ extracted from P_{IJ} or C_{IJ}.

VII. Examples

This section is devoted to examples of edge images produced using various choices for the functions discussed in Sections IV and V. Fig. 14(a) shows a transformed version of the original image we used, which was a digitized mammogram of a normal patient obtained from the database of the *Mammographic Image Analysis Society* (MIAS). The image is identified as "mdb003ll" in the database and is a "large (l)" image in their "smlx" size classification. Interested users can

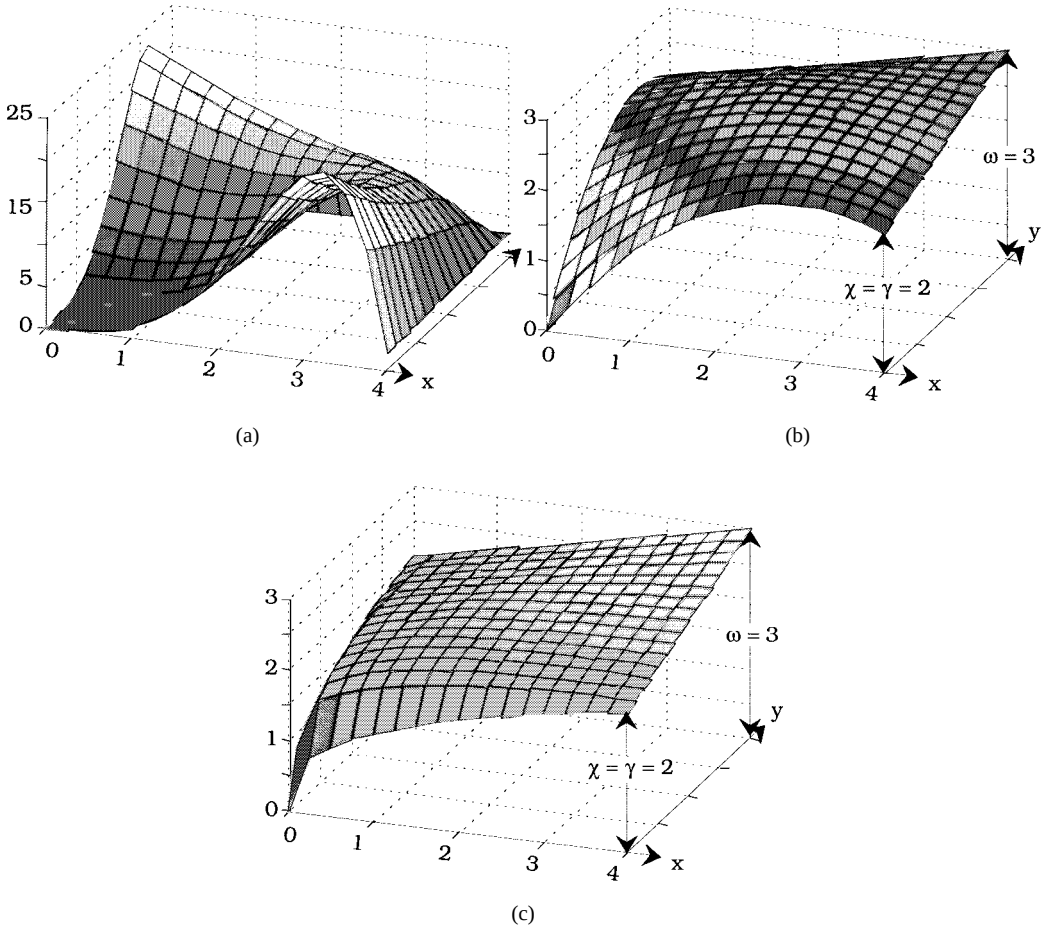


Fig. 13. Three surfaces generated by b_{TS4} on [0, 4] × [0, 4]. (a) $\tau = 4$. (b) $\tau = 1$. (c) $\tau = 0.25$.

obtain further information on the database from the MIAS web site or by e-mailing the MIAS [23].

The original image (called P_{IJ} in Section II) was scanned at 50 μ's per pixel in each of the two orthogonal directions by a scanning densitometer that maps the optical density [$\log_{10}$ (incident light intensity/transmitted light intensity)] to an eight bit number (0–255) at each pixel. The original image size was $m = 2600$ by $n = 4320$. We reduced the original image to 163 (rounded up from 162.5) by 270 pixels (44 010 pixels) at a resolution of 800 μ's per pixel in each direction, by successively averaging 8 × 8 neighborhoods of the original image followed by a second compression of 2 × 2 neighborhoods. This operation was not discussed in Section III, but is a conditioning operation that results in the image in Fig. 14(a). To be consistent, the image in view 14(a) should not be called P_{IJ}; we have done so here to simplify the figure captions.

Following size reduction, we applied the histogram equalization in (4) to image Fig. 14(a), resulting in the conditioned image we call $C_{IJ} = C_{(4)}$ in view 14(b). Edges in image 14(b) have more acuity than those in view 14(a), making visual comparisons with edge images somewhat easier. At the other end of the processing sequence is scaling, represented here as the function $s_{(20)}$ specified by (20). Most of the edge operators we examine have the general form $e = s_{(20)} \circ b \circ \mathbf{f} \circ c_{(4)}$, the exception being experiment 7.2, which studies $s_{(20)}$ itself.

There are some key elements in Fig. 14(b) that we will allude to when comparing different outputs. First, there is the main boundary (or skinline) of the breast and the milky web-like structure within it. Second, there is a very slight area (noise spot) introduced by (4) of just a few pixels to the right of the main boundary and to the left of the number 19. Finally, there is the label box in the upper right-hand corner that surrounds the letters ML. These reference points are especially useful for visual comparisons of different edge images.

Edge images will be discussed that use various combinations of the features $f_{|\mathrm{Sv}|}(\mathbf{w})$, $f_{|\mathrm{Sh}|}(\mathbf{w})$, $f_r(\mathbf{w})$, $f_{\mathrm{sd}}(\mathbf{w})$. Here, $\mathbf{w}$ is a vector of nine intensities from any 3 × 3 window in C_{IJ}, each value of which in the real image can be any integer in $\{0, 1, \cdots, 255\}$. The absolute Sobel features are used here as a convenience: their utility as edge detection features is not importantly different from the signed features $[f_{\mathrm{Sv}}(\mathbf{w}),\ f_{\mathrm{Sv}}(\mathbf{w})]$.

Fig. 5 shows that different features attain different maximum values on the same window. To facilitate unbiased blending of various features, it seems desirable to *normalize* them to a standard range. There are many normalizations. For example, when images have 256 gray levels, we can normalize the absolute Sobel features to the range [0, 255] by division by 4 to agree with the natural range of the $r$sd features. Here, we choose to normalize all features to be real numbers lying in the closed interval [0, 4]. This means that the absolute values

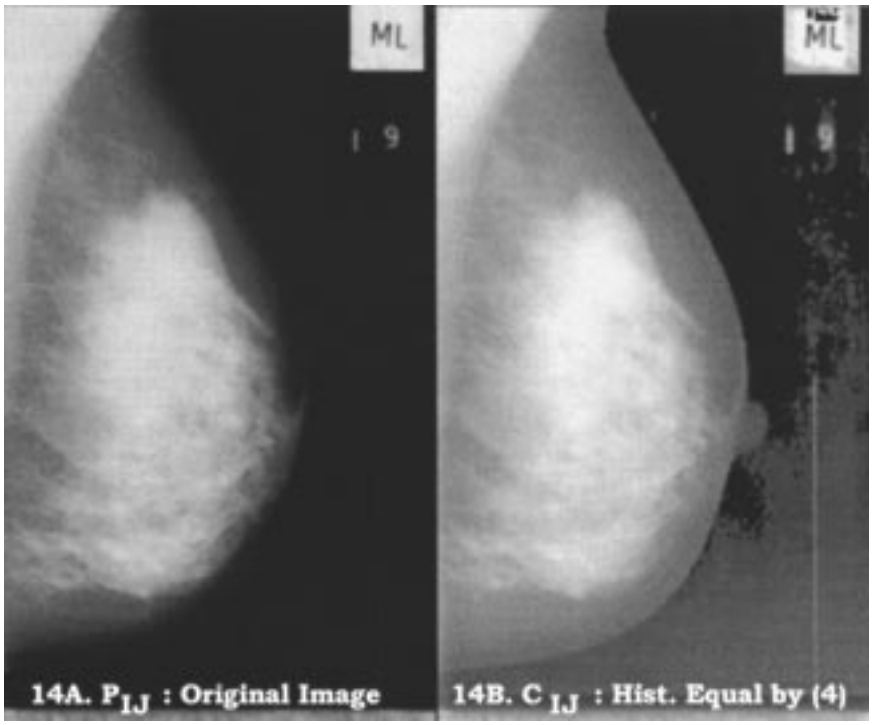


Fig. 14. A (compressed) digitized mammogram and its histogram equalization by (4).

of the Sobel features are divided by 255, r by 63.75 and sd by 31.68, respectively. *We, henceforth, refer to these four normalized features as* h, v, r, *and* s (*not* sd), *respectively*.

The remaining pages of this section discuss various aspects of edge images produced from C_{IJ}. There is no "gold standard" or ground truth edge image that can be used to compare various outputs. Heath *et al.*, [24] advocate an interesting and more rigorous approach to this problem than the usual one (which is to look at them) based on human ratings experiments. Here, we resort to the usual method of side by side visual assessment to compare the efficacy of different models. Before reading our discussion about these images, page through the figures in this section and form your own opinions about their relative visual qualities. For fun, rank the outputs and compare your rankings to ours. Obviously, you may not agree with us, but this exercise may provide you with a little relief from the effort of struggling through the tortuous path that has brought you this far.

Experiment 7.1. Different Features: Fig. 15 shows eight edge images produced by $e = s_{(20)} \circ b_{\|*\|_2} \circ \mathbf{f} \circ c_{(4)}$. The only variable in Fig. 15 is $\mathbf{f}$—the feature extractor function.

First, compare Fig. 15(a)–(d). These four views are made with just one of the four features h, v, r or s. All visible edges of the label box are clearly found using either s or r. On the other hand, h (the normalized absolute horizontal Sobel feature) misses the vertical edges of this box, while its counterpart v misses the horizontal edge of the box. This is expected based on the geometrical meaning of these two features. By contrast, either r or s both provide clear edges in both directions, emphasizing their nondirectional geometric nature.

Fig. 15(c) best shows the structure within the breast that is seen in Fig. 14(b) and, overall, we think it the best of these four views. Gonzalez and Woods state that "the idea underlying most edge-detection techniques is the computation of a local derivative operator" [17, p. 416]. Fig. 15(c) and (d) shows that other kinds of features can also be used to produce good edge images.

Fig. 15(e) and (f) are made with the feature pairs (h, v) and (r, s). Edges in Fig. 15(e) appear somewhat thinner than those in Fig. 15(f) and, as expected, the use of both h and v enables this detector to find all the edges of the ML label box. On the other hand, Fig. 15(f) possesses somewhat more structure within the breast and has somewhat better contrast. The principal difference between these two views can be seen by comparing the letters ML in the label box and the noise pixels in the far right of each view. The gradient-based features (h, v) produce an "edgier" visual effect than the statistically-based (r, s) feature pair. Overall, these two images are really pretty similar.

Panels g and f in Fig. 15 show edge images produced by using the triple (h, v, s) and the whole set (h, v, r, s). The best way to assess each of these views may be to compare each of them to Fig. 15(e), which is the edge image corresponding to (h, v) alone. Adding the standard deviation s to (h, v) as in Fig. 15(g) yields an image that is visually brighter; that is, s seems to enhance contrast, but this image is somewhat more blurred than Fig. 15(e). Adding (r, s) to (h, v) as in Fig. 15(f) gives a result much like Fig. 15(g). All views that used r or s or both produce a more visible noise spot; none of the eight views shows the number 19 well.

We examined many edge images using these and other combinations of features with various blending functions. In many cases, the results were similar, but there were always small and possibly important differences. Fig. 15 should convince you that using nonstandard features as well as

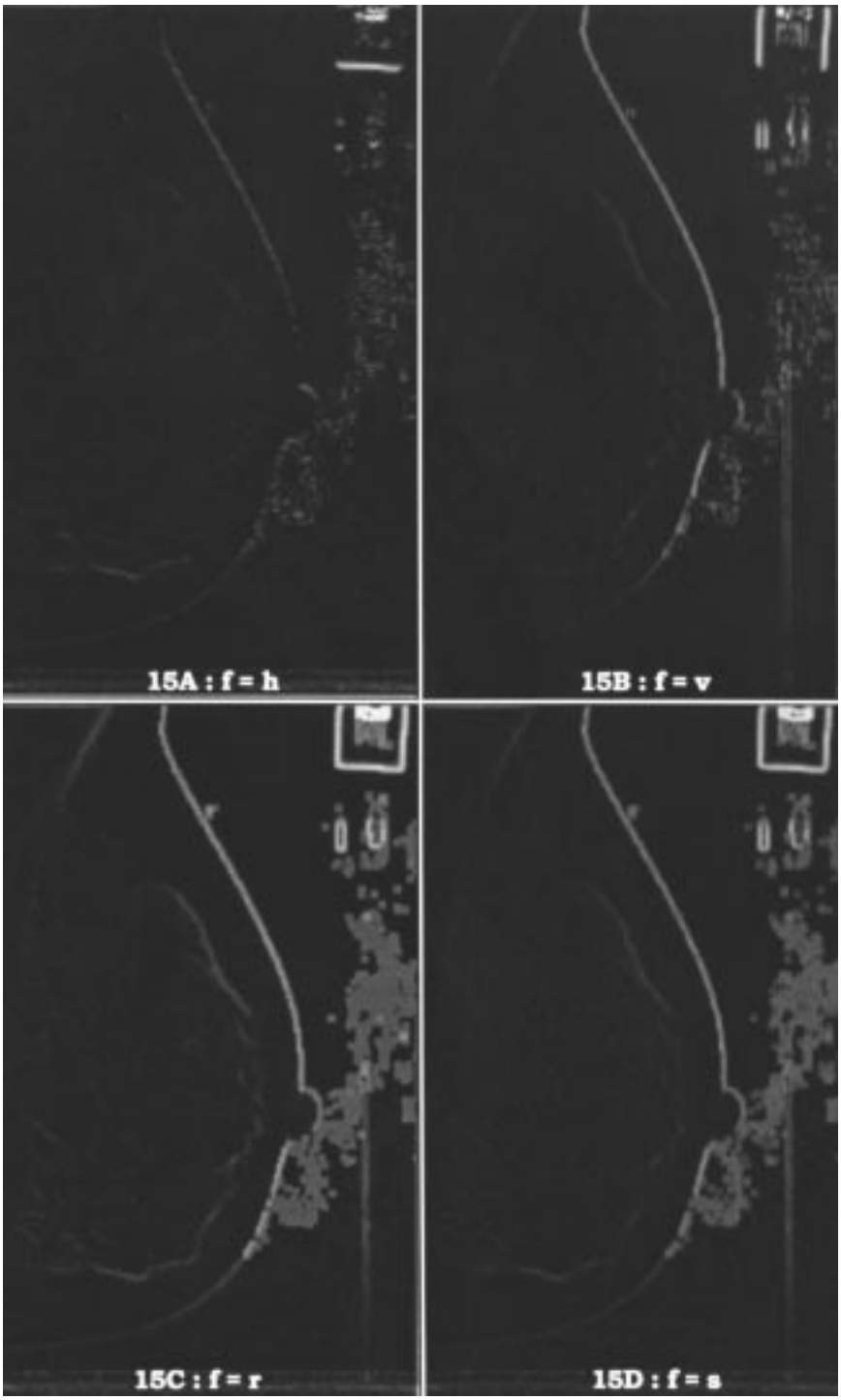


Fig. 15. (a)–(d) Different features using the two norm for blending.

combinations of features such as these—with ANY blending function—may give a result that is better (that is, more useful for the application at hand) than those based on the traditional derivative-like features mentioned by Gonzalez and Woods. Which of these eight views did you rate best? We chose Fig. 15(c), followed by Fig. 15(e) as the second best view.

Experiment 7.2. Different Norm-Based Blending Functions and Scaling: This experiment has two objectives—to compare different norms for $b_{||*||}$ using the features (h, v) and (r, s) and to see the effects of dynamic scaling as done by (20) on images produced by the sequence $b_{||*||_t} \circ \mathbf{f} \circ c_{(4)}$.

Equations (8) and (10) provide two norm families parametrized in A and t. The most obvious candidate from family (8) besides the Euclidean norm at (9) is the Mahalonobis norm induced by the inverse of the covariance matrix of $\mathbf{F}_{IJ}$, $A = [\text{cov}\ (\mathbf{F}_{IJ})]^{-1}$. However, since the Euclidean norm is also in family (10), we chose to illustrate only the one, two, and sup norms in this example.

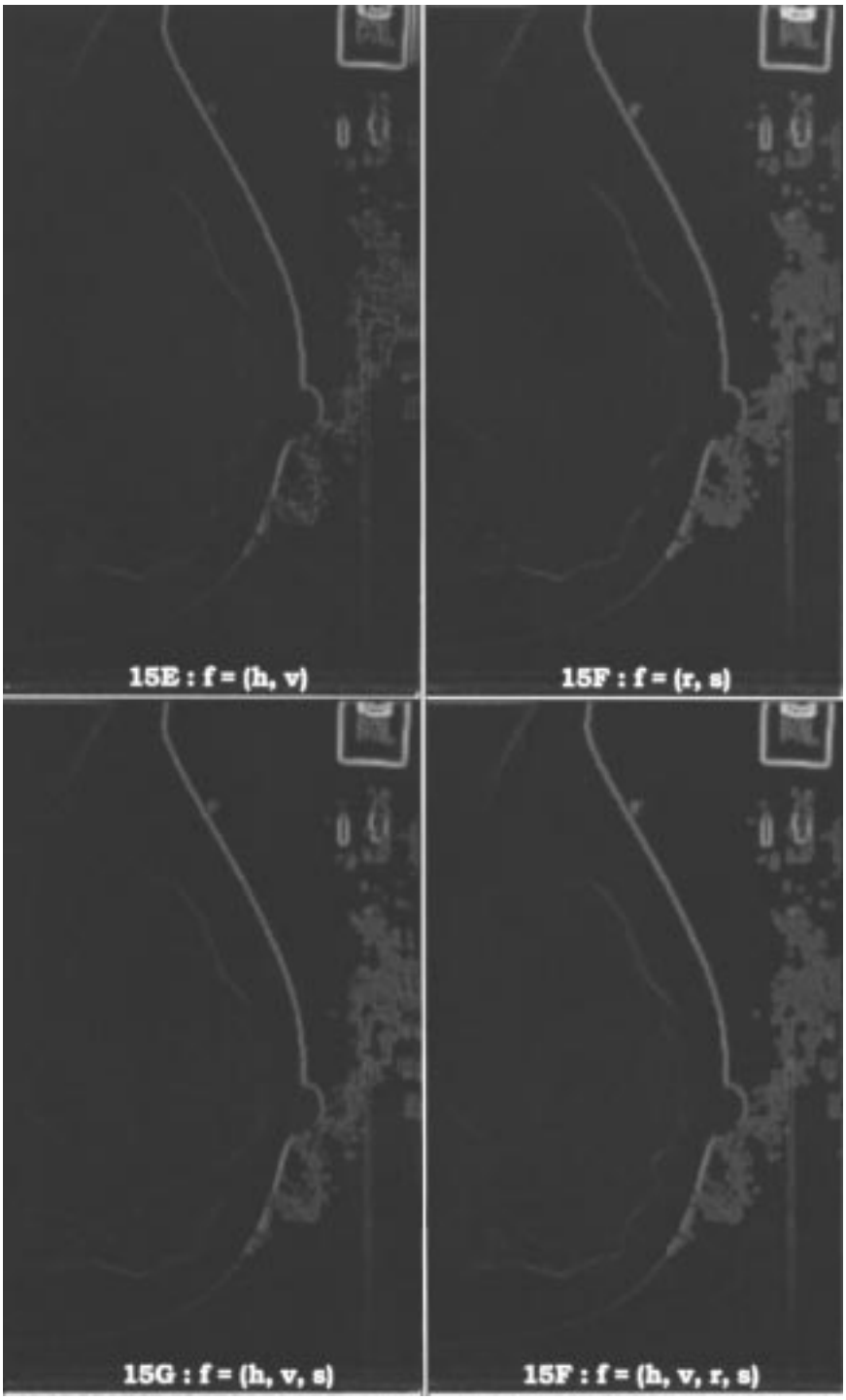


Fig. 15. (*Continued.*) (e)–(h) Different features using the two norm for blending.

Fig. 16 shows four edge images [views (a), (b), (e), and (f)] before dynamic scaling corresponding to $e = s \circ b_{||*||_t} \circ \mathbf{f}_{(h,\,v)\mathrm{or}(r,\,s)} \circ c_{(4)}$, where ($s$) is a constant scale factor chosen so that the one norm would just saturate at 255; and four edge images [views (c), (d), (g), and (h)] after scaling, corresponding to $e = s_{(20)} \circ b_{||*||_t} \circ \mathbf{f}_{(h,\,v)\mathrm{or}(r,\,s)} \circ c_{(4)}$. We use the Minkowski norms for $t = 1$ at (11a) and $t \to \infty$ at (11c) as blending functions. Viewing Fig. 16 in conjunction with parts of Fig. 15 also enables you to compare these two norms with the more familiar Euclidean norm at $t = 2$ (11b). In this and subsequent figures, 1 n, 2 n, and sup n stand for the 1, 2, and sup norms.

Fig. 16(a) and (b) are without scaling and do not have a companion view for the two-norm in Fig. 15. Fig. 16(c) and (d) use the same parameters as Fig. 16(a) and (b), but with dynamic scaling as described in (20). Fig. 15(e) "fits between" these two views as it also uses (h, v), the two-norm, and $s_{(20)}$. As t in (10) increases from one to infinity, edge images produced using (10) for blending become darker and seem to lose definition without dynamic scaling. In contrast, there are

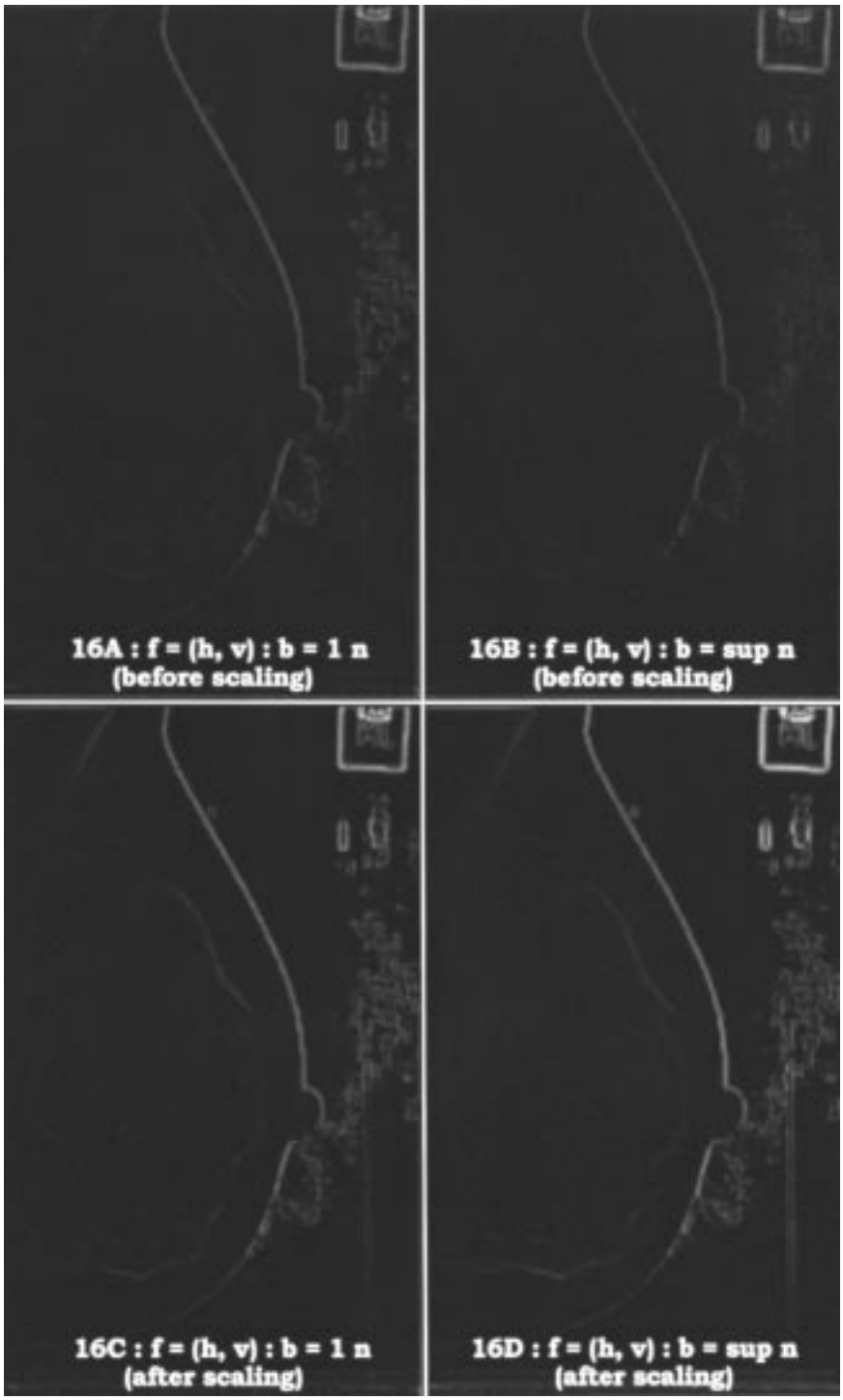


Fig. 16. (a)–(d) Norm blending using (h, v) before and after scaling.

only slight differences between Figs. 16(c), 15(e), and 16(d), corresponding to scaled edge images using the one, two, and sup norms, respectively.

Fig. 16(e)–(h) shows four images with the same scaling and blending functions as 16(a)–(d) using the statistical features (r, s) instead of the normalized absolute Sobel pair (h, v). First, compare the four views 16(e)–(h) to the corresponding views in 16(a)–(d) panel by panel. Using (r, s) brightens all four images and again thickens both edges and noise to some extent. The contrast between the one and sup norms as blending functions is seen more easily in Fig. 16(e) and (f) than in Fig. 16(a) and (b): as t increases, brightness and definition are lost. But we see again in Fig. 16(g) and (h) that scaling essentially removes this difference. Finally, Fig. 15(f) "fits between" these two views as it also uses (r, s), the two-norm, and $s_{(20)}$.

Looking at the before and after views in Fig. 16 shows two things: dynamic scaling enhances edges in the unscaled versions and scaling by (20) also decreases apparent differences in edge quality due to different t-norm blending functions.

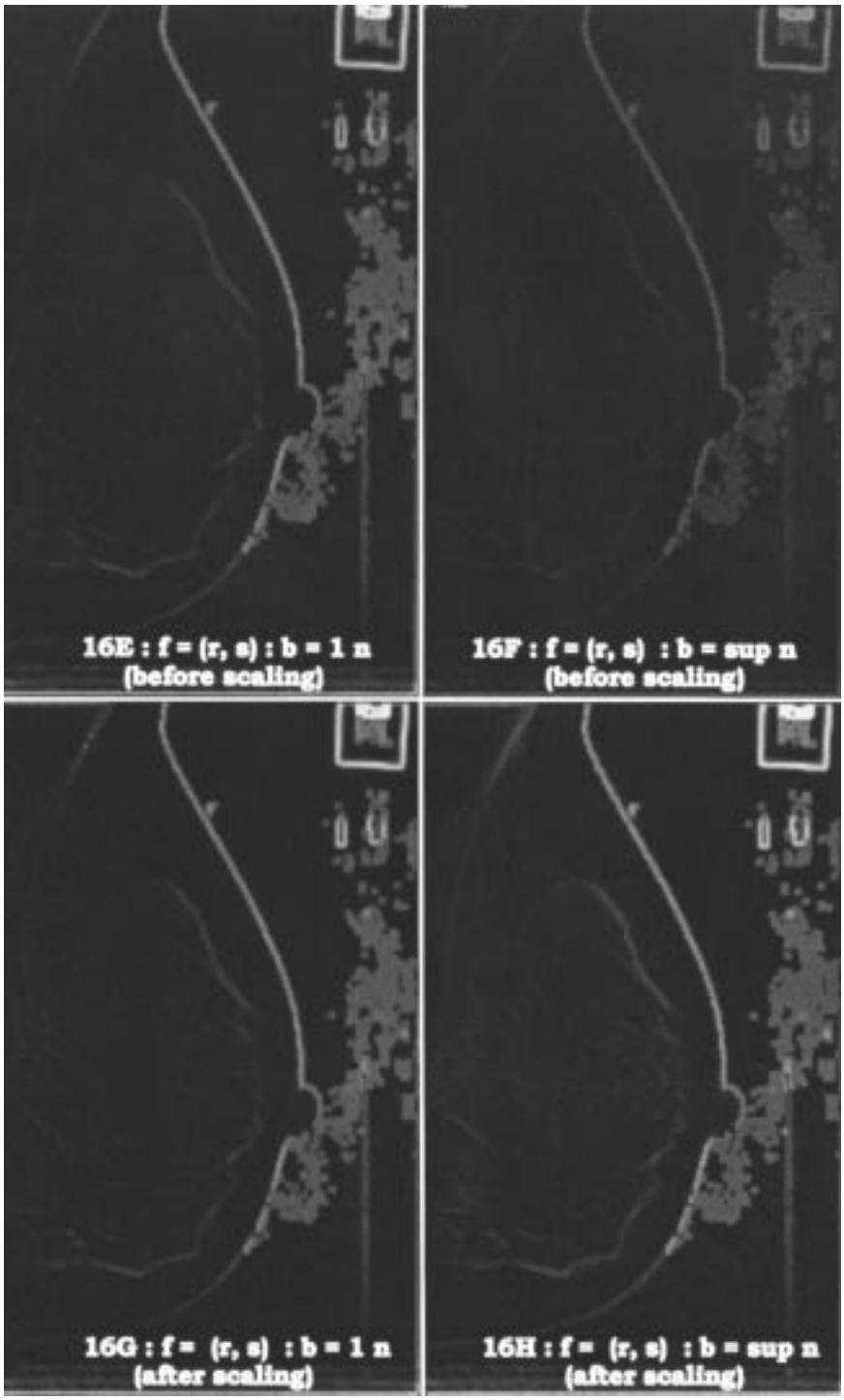


Fig. 16. (*Continued.)* (e)–(h) Norm blending using (r, s) before and after scaling.

An infinite set of edge images can be generated in this way that will be (digitally) continuous in their change from view to view as t runs from one to infinity in (10). It is NOT the case, as asserted in [17, p. 418], that the length of the vector (h, v) in the one-norm is simply an approximation to its length in the two-norm. Every norm gives an equally valid measure of the length of vector (h, v). Mathematically, all norms are admissible and, operationally, it is clearly the case that varying this kind of blending function with all other components of the sequence fixed will result in a wide variety of edge images if care is not taken to use a form of scaling that puts them all on the same relative footing. But if scaling is used appropriately, there seems to be only slight differences in the resultant edge images, so ease of computation might be an important factor in your choice of norm. Which image in Fig. 16 seems most appealing to you? Our pick is Fig. 16(g).

Experiment 7.3. Different Generalized Logistic Blending Functions: Fig. 17 shows eight edge images corresponding

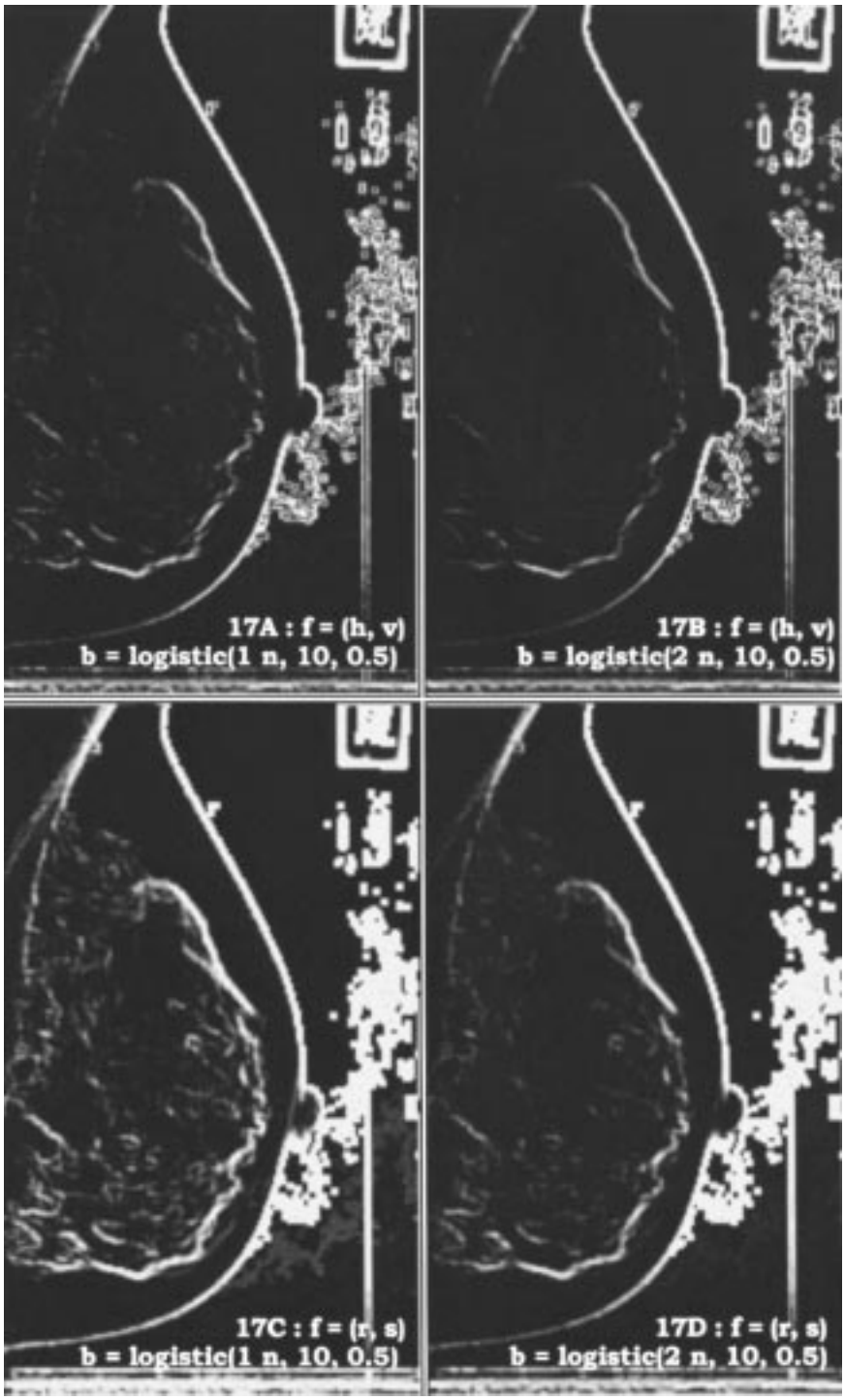


Fig. 17. (a)–(d) Logistic blending functions with $\lambda = 10$ and $\beta = 0.5$ using (h, v) and (r, s).

to $e = s_{(20)} \circ b_L \circ \mathbf{f}_{(hv-\text{or}-rs)} \circ c_{(4)}$. The images in this series use logistic functions as in (12) for blending. There are three parameters for b_L, so it is impossible to show the wide variety of edge images that can be realized as each parameter is varied. Fig. 17 shows a few representative images obtained with this family that enable us to study several facets of logistic blending functions. Fig. 17(a)–(d) compare edge images obtained with λ and β fixed while the norm and features vary. Fig. 17(e)–(h) show images corresponding to a fixed norm and features while λ and β vary.

The only difference between Fig. 17(a) and (b) is the norm used in (12). The one-norm seems to result in somewhat brighter and thicker edges, and the internal structure of the breast is slightly more well-defined than when the two-norm is used. Fig. 17(c) and (d) show the results of using the (r, s) feature pair instead of (h, v) with all other parameters of Fig. 17(a) and (b) fixed.

All four images in Fig. 17(a)–(d) based on logistic blending show more internal structure and thicker edges than previous views based on blending with norms and are, in some loose

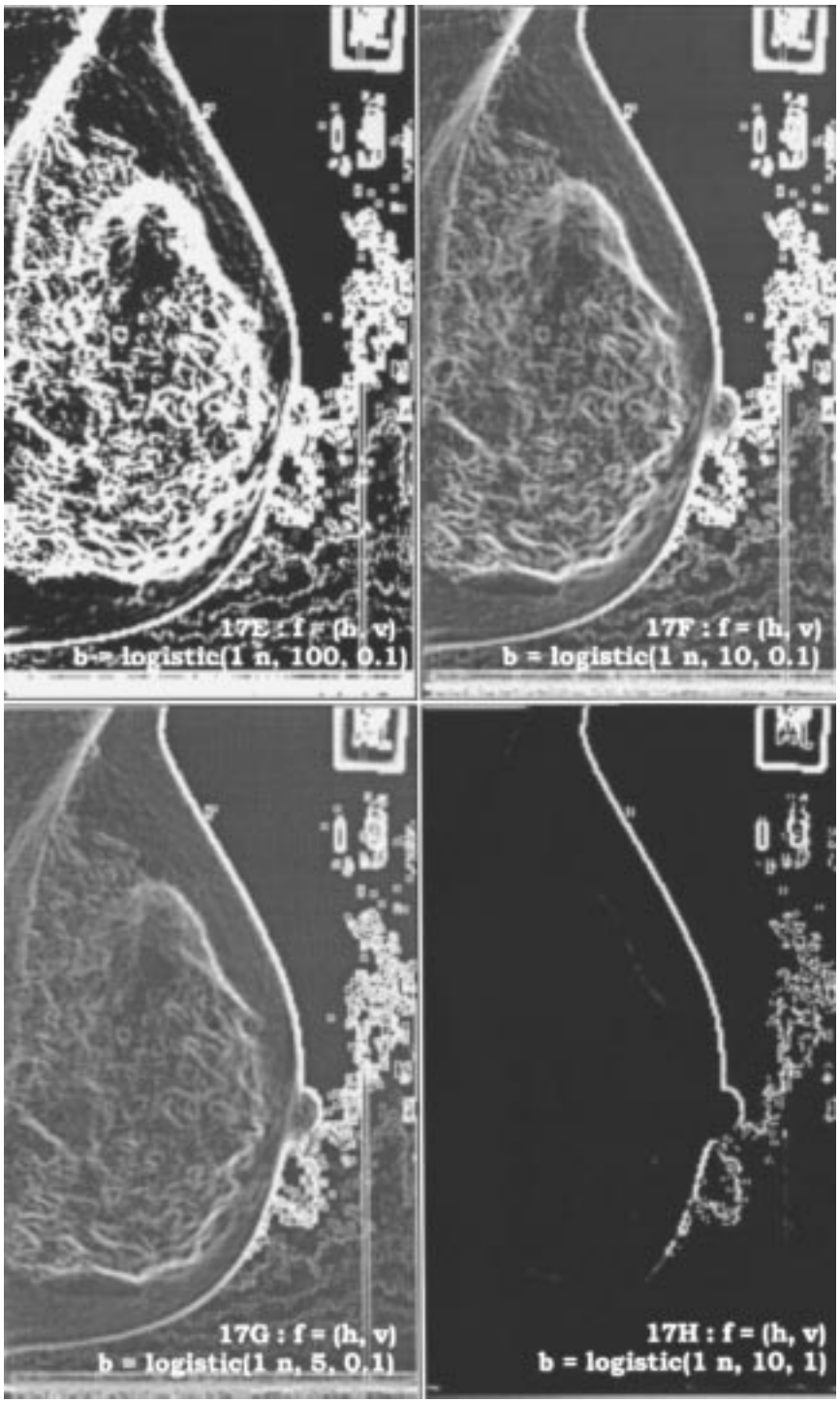


Fig. 17. (*Continued.)* (e)–(h) Logistic blending functions using (h, v) and the one-norm: λ versus β.

sense, "in-between" edge and region segmentations of the image. The (r, s) features reveal many more edges than the normalized Sobel gradients, and the internal structure of the breast is very detailed (refer to Fig. 14(b) for a visual benchmark). And, as in previous figures, (r, s) continues to give brighter and thicker edges than (h, v).

The dependency of b_L on λ and β can be seen in Fig. 17(e)–(h) in which (h, v) and the one-norm in (12) are fixed. Fig. 17(e)–(g) hold $\beta = 0.1$ fixed while λ varies from 100 to 10 to 5. Changing λ has a very pronounced effect on the output. Recall that all images in Fig. 17 are dynamically scaled via (20), so the transition from almost monochromatic at $\lambda = 100$ to very gray at $\lambda = 5$ is directly attributable to the value selected for λ. If we plot b_L as a function of the norm on [0, 4] with λ and β as parameters, we get a series of curves reminiscent of the transfer characteristics of digital switches that switch from one state to another. Parameter β controls the value of the input ($||\mathbf{x}||$) at which the transition

occurs, while λ controls the suddenness or steepness of the transition. When $\lambda = 100$, the transition or switching occurs almost at once and the result is a "digitized" edge image as in panel Fig. 17(e). For this value of λ, b_L behaves as if it were in a "saturation" mode analogous to a transistor switch; it functions as a thresholder and binarizes the image. When $\lambda = 10$ as in Fig. 17(f), the transition is more gradual with a linear, intermediate region. As λ continues to decrease [see Fig. 17(g)], this effect continues until $\lambda \leq 1$, at which the switching behavior is entirely lost and we get "linear" modes with "milky" images.

Finally, compare Fig. 17(f) and (h) (above and below on the right side). The only difference between these two results is that $\beta = 0.1$ in Fig. 17(f), whereas its value in Fig. 17(h) is one. This shows how β can be used to create a range of edge images that appear very gray (small values of β) to essentially black and white (large values of β). The value of β governs when the onset of transition occurs. If β is large [1 or 2, for example, as in Fig. 17(h)], most of the low-intensity edges will be switched out with the result that only the strong edges will show in the final, dynamically scaled image. Making β small [close to zero as in Fig. 17(f)] allows the whole dynamic range of the edge image to be subjected to the sigmoidal modulation of the logistic function. This results in a detail-rich edge image whose appearance depends on λ (crisp black and white if λ is large, and milky if λ is small). If β is made very small and we are in saturation mode, we can get a binary image that roughly segments the breast and background in mammograms. Conversely large β in saturation mode can lead to a totally black image with no information. By choosing λ and β so that b_L operates in the linear mode, we can get an image with the degree of edge detail we desire.

When viewing the panels in Fig. 17, it is useful to remember that for fixed features as in six of the eight Fig. 17 views, variations in the three parameters of $b_L(\mathbf{x}; \|*\|, \lambda, \beta)$ change the shape of the graph of the blending surface corresponding to a particular function. Fig. 17 shows that logistic blending functions provide a very rich family of edge detectors that yield an almost infinite variety of edge images. The underlying mathematical reason for this lies with the shape of the graphs that correspond to different members of the blending function family. Incidentally, we would, as with previous figures, certainly call all of the images in Fig. 17 that use the (h, v) feature pair *Sobel edge images*. Which of these images do you prefer? For this example, it is hard to choose, because there are really several types of images. Our pick for the best edge definition is Fig. 17(a), and the best internal structure seems to be in Fig. 17(f).

Experiment 7.4. Different TS Blending Functions: Fig. 18 shows six edge images for $e = s_{(20)} \circ b_{\mathrm{TS4}} \circ \mathbf{f}_{(h, v) \text{ or } (r, s)} \circ c_{(4)}$. There are four parameters for b_{TS}, so it is again impossible to show the wide variety of edge images that can be realized as each of these is varied. To study our contention that the behavior of the TS4 blending function near the origin is more important than elsewhere, we study variation in e using b_{TS4} only as a function of τ in (18a) and the features it blends together. Panels (a), (b), and (c) in Fig. 18 use the absolute Sobel features (h, v), whereas views (d), (e), and (f) use the (r, s) pair. Each vertical block uses the same choices for b_{TS}, so this affords a third comparison of edge image quality as a function of extracted features. The images in this series are all dynamically scaled with (20).

Fig. 18(a)–(c) [and also Fig. 18(d)–(f)] correspond to the TS4 blending functions having the shapes of the three surfaces shown in Fig. 13(a)–(c), respectively. In particular, for $\tau = 4$ [Fig. 18(a) and (d)] the blending function has the graph shown in Fig. 13(a)—that is, b_{TS4} is locally convex. Because this blending function shape is locally similar to the two-norm (the four norm would make a better comparison, but we have not shown images in Fig. 15 and 16 for the 4 norm), this pair of images is best compared to Fig. 15(e) and (f), the analogous edge images produced on these two feature sets with the two-norm as the blending function. There, as here, the edge images lack contrast and structural detail; and in both places, the (r, s) feature pair seems to produce brighter, thicker edges.

The middle two frames, Fig. 18(b) and (e) for $\tau = 1$ show edge images that are best compared to Fig. 16(c) and (g), respectively, both of which use the one-norm. Match Fig. 16(c) to 18(b): we think that Fig. 18(b) is much brighter and more structurally detailed. Now compare Fig. 16(g) to 18(e); again, the TS4 image appears quite superior to its counterpart using the one-norm for blending. In our opinion, Fig. 18(b) is a much crisper and more detailed edge image than any shown so far. Fig. 18(c) and (f) shows the result for $\tau = 1/4$ with the two feature pairs. At this value of τ the graph of b_{TS4} ramps up quite steeply (perhaps *too* steeply for good *edge* detection) for feature vectors near $\mathbf{0}$. This pair of images are most comparable visually to the milky, segmentation-like outputs produced by the logistic function—Fig. 17(f) for example.

Comparing views in Fig. 18(a)–(c) [or Fig. 18(d)–(f)] imparts the striking effect that variation in the shape of the graph of the blending function near the origin can have as τ progresses from 4 to 1/4. Moreover, this also shows that the TS4 edge detector, like logistic blending functions, can produce a wide and richly diverse variety of edge images as a function of its parameters. The image in Fig. 18(e) might be useful for detection of the skinline, whereas the image in Fig. 18(f) might find applications in the characterization and detection of mass lesions. Views in Fig. 18(e) and (f) are our choices for the most visually informative images in this series. Which did you pick?

VIII. Conclusions and Discussion

We have made and will briefly discuss six points, viz.:

1) viewing edge detection via the architecture shown in Fig. 2 clarifies the role of each part of the edge detection process;
2) good features for edge detection need not be digital gradients but may be statistical, etc.;
3) a small basis set of windows may clarify feature performance and are essential for training computational learning models for edge detection;
4) a proper match between the desired features of the edge detector and the graph of the blending function leads to (visually) optimal edge images;

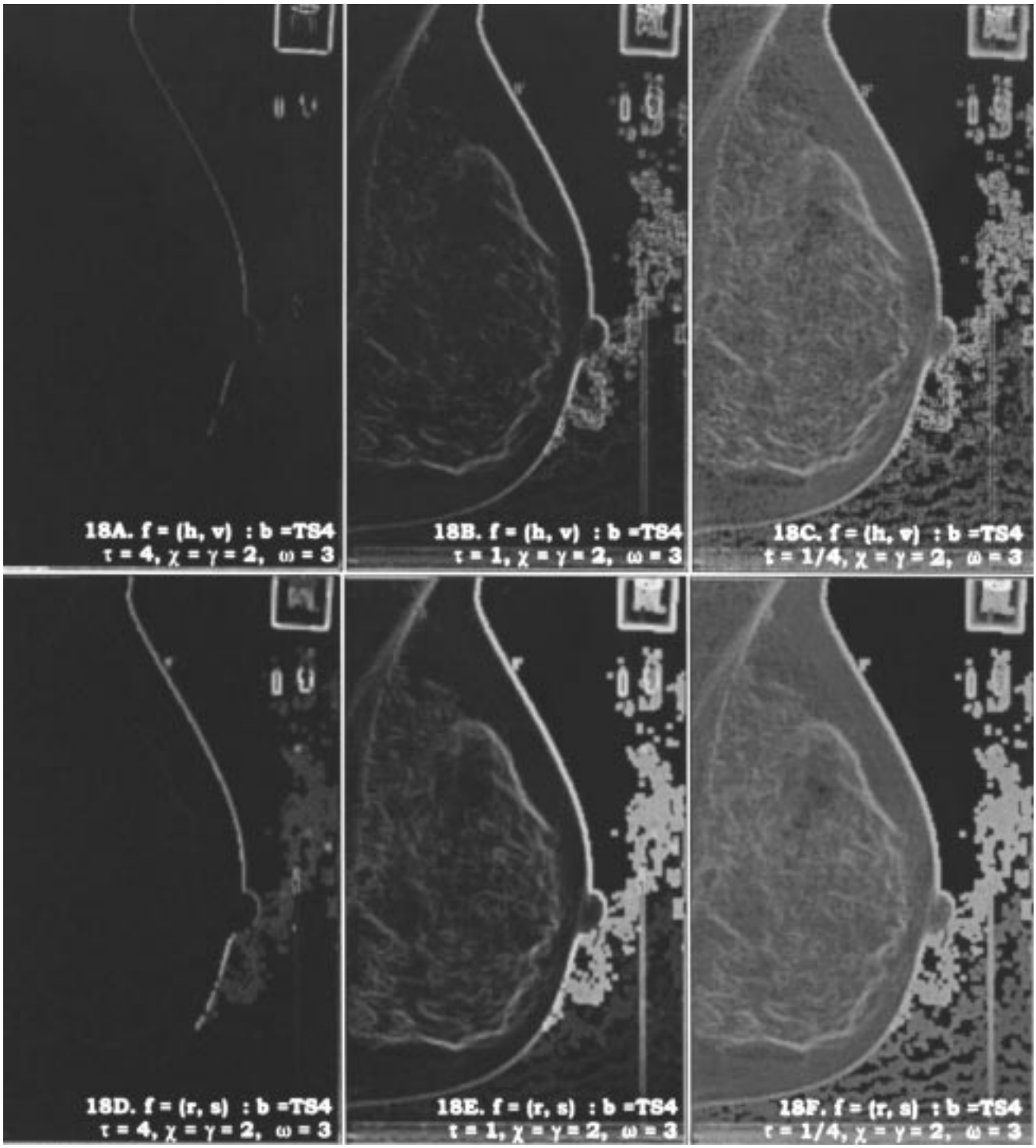


Fig. 18. TS4 blending functions using (h, v) or (r, s) with $\chi = \gamma = 2$ and $\omega = 3$.

5) blending functions whose graphs have the (loosely characterized) waterfall function shape may provide better edge response than norms;
6) parametric families of blending functions such as the generalized logistic functions and the Takagi–Sugeno model provide a means for generating a richly diverse set of edge images through control of simple parameters.

The first point we emphasize is that our architecture clarifies the role played by the features in edge detection models. The Sobel masks in (5) produce edge related features which are fixed in value for a given image. Our decomposition of the detection process makes it clear that there is not just one Sobel edge image, but infinitely many. It is entirely proper, in our view, to call every image in Fig. 15–18 that are based entirely on h or v or (h, v) a Sobel edge image. There are as many different Sobel images as you care to construct and they may have very different visual appearances and utility.

Another important point that has been made is that the features chosen need not be entirely composed of estimates of

digital gradients. Gradient-like features based on the geometry of derivatives have three important properties (direction of maximum rate of change, maximum magnitude of the change, and digital orthogonality) for edge detection. Our examples have shown that the range and standard deviation of pixel intensities in $\mathbf{w}$ can serve equally well, and in some cases, better than estimates of digital gradients. $r$sd features have four important properties that complement digital gradients: they are rotationally invariant, nonorthogonal, invariant (in functional form) to window size, and have well known statistical distributions. We have also shown that there is no good reason to constrain the search for edge features to 2-D derivative-based vectors and there is no *a priori* reason not to try combinations of various features, as long as each possesses edge detecting capabilities via its geometric meaning. Moreover, the analysis of features that may or may not be useful for edge detection is greatly enhanced by using a small and easily understood set of basis windows such as B_{512}.

Once the features are chosen, we argued that the choice of a blending function should not rest mainly with historical precedent or computational convenience. We have exhibited three families of blending functions that all produce viable edge images. We believe that the key to designing a useful edge detector for a particular application should begin with a careful analysis of desirable properties of the edge image and subsequent choice of an appropriate blending function. How to do this for more than two features is an open question. Another interesting problem that deserves future research is a careful mathematical specification of the properties that make a blending function have a waterfall surface.

Another important step taken in this paper was to show how to use fuzzy input–output rule-based systems such as the Takagi–Sugeno model for edge detection; papers [11]–[15] all use the Mamdani [20] fuzzy system. This can be done in one of three ways: 1) by *specification* of system parameters (as we did for b_{TS4}); 2) by *derivation* of optimal coefficients (as was done in [16] for b_{TS12}); or 3) by *training* a computational learning model using $\mathcal{I}_{\mathrm{IO}}(B, \mathbf{f}, b)$ (as was done in [10] for b_{NN}). Further, we showed how the geometric approach could be used to get model-based IO training data that are needed for learning when method 3) is used. There are many things that can be studied in connection with the TS model. For example, the configuration of the LHS of the rule base affects the edge response and offers something akin to compartmentalized tuneability. The sensitivity of b_{TS} depends on the choice of membership functions for each linguistic variable. Although symmetric triangular functions are convenient, it would be interesting to study how to optimize the number (granularity) and shape of the membership functions that comprise the termsets for each input variable. The firing strength of each rule depends on the T-norm used for intersection. Here we used $\mathcal{T}_2$. There are seven infinite families of $\mathcal{T}$-norms and several families of averaging operators that can be used instead [20]. A change in this parameter clearly affects the edge image produced. The training set B_{512} has obvious drawbacks. B_{512} can be enriched in a number of ways. For example, windows with values between zero and one (0.5 corresponds to a gray level of 127) can be added to it.

The application domain hinted at by the examples in this paper is the analysis of digital mammograms. Information about intensity, edges, texture, and shape plays a central role in image segmentation and lesion detection in mammograms [25]–[28]. The results of this study are important to our application in at least three ways.

1) *Feature selection*—edge and texture are generally regarded as complementary properties of an image. We have shown in this study that the standard deviation of pixel values within a window—a traditional texture feature—can function as a feature for edge detection as well [33]. The judicious and economic choice of features that embody "multiple dimensions of information" (such as edge and texture) is vital to insightful characterization and successful detection of lesions.
2) *Adaptive blending of features*—the need for local processing in lesion segmentation is well known [28], [29]. Indeed, it is common for a single, continuous edge in a mammogram (such as the skinline of the breast) to vary in strength along its length [see for example, the bottom fifth of Fig. 15(e)]. A tuneable edge detector that strengthens weak edges without emphasizing strong ones is desirable for automatically extracting these edges. The parametrized analytical (logistic) and computational (TS4) blending function families introduced in this study (Figs. 17, 18) allow tuneable edge detectors to be designed systematically. Thus, one obvious application of our results is to delineate the skin-air interface, the pectoral muscle boundary and the extent of the glandular tissue on a mammogram [Figs. 17(f), 18(c), 18(f)]. Such blending functions are also ideal for blending intensity, edge and texture features adaptively to detect subtle mass lesions, for example.
3) *Interactive optimization of diagnostic systems by practicing clinicians*—an exciting possibility for computational mammography is on-line tuning of digitally enhanced mammograms by practicing clinicians. Blending functions such as b_L and b_{TS} offer a simple and convenient means for on-line visual optimization by adjustments to simple parameters of the functions. Thus, different features in an image can be enhanced in near real time by simply "turning a dial." We will be participating with a local hospital in experiments to ascertain the usefulness of this idea in the near future.

Finally, it would be interesting to pursue the perceptual mechanism that underlies the cognitive process of edge detection by humans implied by this study. It was our observation in this article that more than linear (waterfall) response to edge features near the origin of feature space coupled with a flat response far from $\mathbf{0}$ seemed to produce visually better edge images than norms (which have less than linear response near $\mathbf{0}$ and which increase monotonically as feature vectors get further from the origin). Our conjecture is that humans perceive edges with only slight changes in the features they gather and that edges are lost as intensities approach saturation

levels. However, this is a study that is beyond our present capabilities: we hope that one of our readers finds it interesting enough to pursue.

REFERENCES

[1] V. S. Nalwa, *A Guided Tour of Computer Vision.* Reading, MA: Addison-Wesley, 1993.

[2] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Royal Soc.*, vol. B-207, pp. 187–217, 1980.

[3] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.* vol. PAMI-8, no. 6, pp. 679–698, 1986.

[4] J. J. Shen and S. S. Castan, "An optimal linear operator for step edge detection," *CVGIP: Graphical Models and Image Processing,* vol. 54, no. 2, pp. 112–133, 1992.

[5] P. Meer, S. Wang, and T. Wechsler, "Edge detection by associative mapping," *Pattern Recognition,* vol. 22, no. 5, pp. 491–503, 1989.

[6] H. McCauley, "Target cueing—A heterogeneous neural network approach," in *Applicat. Artificial Neural Networks, SPIE Proc. 1469,* Bellingham, WA, 1991, pp. 69–76.

[7] S. Weller, "Artificial neural net learns the Sobel operators (and more)," *Applicat. Artificial Neural Networks, SPIE Proc. 1469,* Bellingham, WA, 1991, pp. 219–224.

[8] C. T. Tsai, Y. N. Sun, P. C. Chung, and J. S. Lee, "Endocardial boundary detection using a neural network," *Pattern Recognition,* vol. 26, no. 7, pp. 1057–1068, 1993.

[9] S. Lu and A. Szeto, "Hierarchical artificial neural networks for edge enhancement," *Pattern Recognition,* vol. 26, no. 8, pp. 1149–1163, 1993.

[10] J. C. Bezdek and D. Kerr, "Training edge detecting neural networks with model-based examples," in *Proc. 3rd IEEE Int. Conf. Fuzzy Syst.*, Piscataway, NJ, 1994, pp. 894–901.

[11] T. Law, H. Itoh, and H. Seki, "Image filtering, edge detection, and edge tracing using fuzzy reasoning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 5, pp. 481–491, 1996.

[12] C. Tyan and P. Wang, "Image processing—Enhancement, filtering, and edge detection using the fuzzy logic approach," in *Proc. 2nd IEEE Conf. Fuzzy Syst.*, Piscataway, NJ, 1993, pp. 600–605.

[13] J. C. Bezdek and M. Shirvaikar, "Edge detection using the fuzzy control paradigm," in *Proc. 2nd Eur. Congress Intell. Tech. Soft Computing*, Aachen, Germany, 1994, vol. 1, pp. 1–12.

[14] C. W. Tao, W. E. Thompson, and J. S. Taur, "A fuzzy if–then approach to edge detection," in *Proc. 2rd IEEE Int. Conf. Fuzzy Syst.*, Piscataway, NJ, 1993, pp. 1356–1361.

[15] F. Russo and G. Ramponi, "Edge extraction by FIRE operators," in *Proc. 3rd IEEE Int. Conf. Fuzzy Syst.*, Piscataway, NJ, 1994, p. 249.

[16] J. C. Bezdek, F. M. Cheong, T. Dillon, and D. Karla, "Edge detection using fuzzy reasoning and model-based training," in *Computational Intelligence: A Dynamic System Perspective,* M. Palaniswami, Y. Attikiouzel, R. J. Marks, D. Fogel, and T. Fukuda, Eds. Piscataway, NJ: IEEE Press, 1995, pp. 108–125.

[17] R. Gonzalez and R. Woods, *Digital Image Processing.* Reading, MA: Addison-Wesley, 1992.

[18] S. Haykin, *Neural Networks: A Comprehensive Foundation.* New York: Macmillan, 1994.

[19] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 1, pp. 116–132, 1985.

[20] H. T. Nguyen, M. Sugeno, R. Tong, and R. Yager, *Theoretical Aspects of Fuzzy Control.* New York: Wiley, 1995.

[21] S. Smith and C. Comer, "Automated calibration of a fuzzy logic controller using a cell state space algorithm," *IEEE Contr. Syst. Mag.*, Aug. 1991, pp. 18–28, 1991.

[22] R. J. S. Jang, "ANFIS: Adaptive-network-based fuzzy inference system, *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, 1993.

[23] Mammographic Image Analysis Society, (MIAS), digital mammogram database, published electronically, 1994; mias@sv1.smb.man.ac.uk (e-mail); http://s10d.smb.man.ac.uk/MIAScom.html (WorldWideWeb).

[24] M. Heath, S. Sarkar, T. Sanocki, and K. Bowyer, "Comparison of edge detectors: A methodology and initial study," in *Proc. 1996 IEEE Comput. Vision Pattern Recognition,* Piscataway, NJ, to be published.

[25] W. P. Kegelmeyer, "Computer detection of stellate lesions in mammograms," in *SPIE Proc. 1660, Biomed. Image Processing 3-D Microscopy,* SPIE, Bellingham, WA, 1992, pp. 446–454.

[26] J. Dengler, S. Behrens, and J. F. Desaga, "Segmentation of microcalcifications in mammograms," *IEEE Trans. Med. Imaging,* vol. 12, pp. 634–642, 1993.

[27] R. Gupta and P. E. Undrill, "The use of texture analysis to delineate suspicious masses in mammography," *Phys. Med. Biol.*, vol. 40, pp. 835–855, 1995.

[28] N. Petrick, H.-P. Chan, B. Sahiner, and D. Wei, "An adaptive density-weighted contrast enhancement filter for mammographic breast mass detection," *IEEE Trans. Med. Imaging,* vol. 15, pp. 59–67, 1996.

[29] M. Kallergi, K. Woods, L. P. Clarke, W. Qian, and R. A. Clark, "Image segmentation in digital mammography: Comparison of local thresholding and region growing algorithms," *Computerized Med. Imaging Graphics,* vol. 16, pp. 323–331, 1992.

[30] J. Varga and L. T. Koczy, "Explicit formulae of two input fuzzy control," *Bull. Ensembles Flous Applicat.*, vol. 63, pp. 58–66, 1995.

[31] L. T. Koczy and M. Sugeno, "Explicit formulae for fuzzy control systems and the approximator property," LIFE TR 93-94/408, LIFE, Yokohama, Japan, 1994.

[32] A. El Hajjaji and A. Rachid, "Explicit formulae for fuzzy controller," *Fuzzy Sets Syst.*, vol. 62, pp. 135–141, 1994.

[33] A. K. Jain, *Fundamentals of Digital Image Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1989, p. 345.

Jim (Steelhead) Bezdek (M'80–SM'90–F'92) received the Ph.D. from Cornell University, Ithaca, NY, in 1973.

His interests include pattern recognition, fishing, computational neural networks, snow skiing, image processing, blues music, medical computing, and motorcycles.

Dr. Bezdek is the Founding Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS.

Ramachandran Chandrasekhar (M'97) received the B.Eng. degree (first-class honors) in electronic engineering from the University of Western Australia, Perth, in 1976, the Master of Applied Science degree by the University of Toronto, ON, Canada, in 1982, and the Ph.D. in philosophy (with distinction) by the University of Western Australia, in 1997.

He is a member of the editorial board of the journal *Health Devices*. He has been with the Singapore General Hospital since 1976 and currently heads the Biomedical Engineering Department there. His research interests include clinical engineering, medical instrumentation design and safety, image processing, and pattern recognition.

Dr. Chandrasekhar is a member of the Association for the Advancement of Medical Instrumentation (AAMI).

Yianni Attikiouzel (M'74–SM'90–F'97) received the B.Sc. (electrical engineering, first-class honors) and the Ph.D. degrees from the University of Newcastle-upon-Tyne, U.K., in 1969 and 1973, respectively.

He is Professor of Electrical and Electronic Engineering at The University of Western Australia, Perth, and Director of the Centre for Intelligent Information Processing Systems in the Department of Electrical and Electronic Engineering, at the same university. He has been active in the areas of adaptive signal processing, information technology, medical electronics, and artificial neural networks. His work has been published in over 200 refereed papers in international journals and conferences. He is the author of two books and he recently co-edited an IEEE-Press book on computational intelligence.

Dr. Attikiouzel is a Fellow of the IEE and IE Australia.